Universidad Zaragoza
1542

Trabajo Fin de Grado

Infraestructura tecnológica para la extracción de recursos de información agrarios disponibles en Internet

Technological infrastructure for harvesting agricultural information resources available on the Internet

Autor

Alejandro Gutiérrez Bolea

Directores

Javier Lacasta Miguel

Ingeniería Informática

Escuela de Ingeniería y Arquitectura

2019/2020

Infraestructura tecnológica para la extracción de recursos de información agrarios disponibles en Internet

RESUMEN

Son considerados datos abiertos todos aquellos datos accesibles y reutilizables, sin exigencia de permisos específicos. En la actualidad, se está produciendo constantemente la publicación de datos abiertos, fundamentalmente por parte de administraciones públicas, siendo el ámbito agropecuario una zona donde hay multitud de recursos de información que se publican periódicamente en la web ajustados a diferentes formatos y modelos. Esta periodicidad supone que determinados días de la semana o del mes, aparece un nuevo conjunto de datos disponible en la Web.

El desarrollo de este Trabajo Fin de Grado (TFG) tiene como objetivo principal crear una infraestructura que posibilite la extracción periódica de recursos de información agropecuarios publicados en Internet, y su integración en un data lake (o data hub). El proyecto hace hincapié en la generalización de los procesos de Extracción, Procesado y Carga (ETL) de cara a permitir la adición de nuevos recursos según éstos se vayan identificando. Para ello, se ha realizado una identificación de patrones en las fuentes de datos para el desarrollo de procesos de extracción generalistas por patrones, permitiendo que se puedan incorporar nuevos lectores de manera directa.

El sistema desarrollado se ha validado con la integración de dos recursos distintos como es el porcino y los cereales y con el desarrollo de una aplicación web que permite la planificación de los trabajos ETL, y otra que muestre las posibilidades de explotación.

La aplicación directa de este software es la extracción de diferentes recursos para su tratamiento, análisis y fácil visualización, permitiendo su posterior integración en otras herramientas como el uso de Inteligencia Artificial para el establecimiento del precio de referencia del porcino.

DECLARACIÓN DE AUTORÍA

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Alejandro Gutiérrez Bolea ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Ingeniería Informática (Título del Trabajo)

Infraestructura tecnológica para la extracción de recursos de información agrarios disponibles en Internet

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24 de Junio de 2020

Fdo: Alejandro Gutiérrez Bolea

ÍNDICE

1. INTRODUCCIÓN

1.1 Contexto del Trabajo

Son considerados datos abiertos todos aquellos datos accesibles y reutilizables, sin exigencia de permisos específicos. Los datos abiertos están centrados en material no documental como información geográfica, el genoma, compuestos químicos, fórmulas matemáticas y científicas, datos médicos, biodiversidad, etc (Wikipedia, 2020). En la actualidad, se está produciendo constantemente la publicación de datos abiertos, fundamentalmente por parte de administraciones públicas.

Dentro del ámbito agropecuario hay multitud de recursos de información que se publican periódicamente en la web ajustados a diferentes formatos y modelos. En este proyecto se han extraído diversos recursos como los precios del porcino y de los cereales como el arroz, cebada, maíz y trigo. Dichos recursos son publicados en Informe Semanal de Coyuntura en la página web del Ministerio de Agricultura, Pesca y Alimentación (Gobierno de España, 2020). En el mismo sitio web también son subidas una serie de publicaciones sintetizando las operaciones estadísticas sobre dichos datos, como son el Anuario de Estadística, fruto de la recopilación de datos de diversas fuentes, internas y externas al Ministerio, con el objetivo de integrar en una única publicación la información de mayor relevancia para los diferentes sectores que integran el Departamento, y el Boletín Mensual de Estadística, donde se presenta una recopilación de principales indicadores económicos, así como un amplio conjunto de indicadores relacionados con el sector agrario, ganadero, pesquero y desarrollo rural entre otros. Además de estos recursos, se han incluido a la hora de la extracción publicaciones de datos a nivel europeo, más concretamente el Informe Semanal de precios de los canales de cerdos (Clase S, E y R) y los precios de los lechones en Europa (Commission, 2020).

En el mercado agropecuario, el gobierno mantiene un histórico de los precios de los productos básicos. Dichos históricos son usados para analizar la evolución de los precios de bienes de consumo básico y tienen una directa relación con el consumo en los hogares y el precio de vida medio como se refleja en los informes. En agricultura se da el hecho de que se van publicando nuevas versiones de los datos continuamente, pero a su vez se mantienen las anteriores porque puede ser de gran ayuda a la hora de analizar y comparar este tipo de datos, ya que son series históricas de información.

Para poder trabajar con los datos, es necesario extraerlos a una base de datos propia que permita operar con ellos. Tener acceso a datos oportunos es imprescindible para tomar mejores decisiones y realizar operaciones comerciales sin problemas, debido a esto, muchas empresas dependen de la extracción de datos por lotes. Un ejemplo de ello sería el uso de recursos de información agropecuarios para la integración de técnicas de Inteligencia Artificial para el establecimiento del precio de referencia de la carne de porcino. Además, se trata de un proceso que conlleva más dificultad por el hecho de que se tenga que ir manualmente a buscar los datos cada vez que se publican y para ello es posible crear automatismos. El mercado agropecuario tiene un contexto histórico y manual. Incluso actualmente, la automatización es escasa lo que dificulta la estandarización de los datos. La aplicación de la automatización aporta ciertas ventajas al usuario

como la comodidad, la seguridad y la adaptabilidad y además se traduce en un ahorro económico y temporal. Esta creación de automatismos es muy útil, pero también hace que nos enfrentemos con tres posibles problemas:

- La variación de los formatos de publicación que obliga a ir monitoreando continuamente los procesos de extracción de datos y hacer modificaciones en los componentes. Por ejemplo, en el desarrollo de este proyecto se ha dado esta situación con la extracción de datos del porcino, ya que en el año 2020 los precios de la carne de cerdo pasaron de publicarse de la hoja 10 a la hoja 12, del documento Excel que los expone en la Web. Aunque pueda parecer un cambio menor, es más que suficiente para que un sistema automático deje de funcionar correctamente.
- La desaparición de fuentes de datos que tiene un problema parecido al de la variación, pero que no lleva la necesidad de reprogramar sino de cancelar los automatismos. No obstante, previamente habremos tenido que diagnosticar la naturaleza del problema que hacía que nuestro automatismo no funcionará. En ocasiones este diagnóstico no es obvio.
- La aparición o descubrimiento de nuevas fuentes de información. Dichos recursos, se deben incorporar para poder tener los sistemas de explotación operativos y competitivos. Esto lleva asociado el desarrollo de nuevos componentes de extracción si no es posible reaprovechar los ya existentes. Un problema previo es la toma de conciencia de la existencia de un nuevo recurso de información. Como cabe esperar, la disponibilidad de estos nuevos recursos no es anunciado o, si lo es, el anuncio se produce en foros profesionales muy específicos.

Este proyecto se integra en las líneas de trabajo del grupo de investigación de Sistemas de Información Avanzados (IAAA, 2020). Concretamente en lo referente al análisis y explotación de recursos de información publicados en la web en el ámbito agropecuario. Un ejemplo de estos trabajos es el mencionado proyecto de uso de técnicas de inteligencia artificial para proponer precio de lonja de la carne de porcino.

1.2 Motivación y descripción del problema

Cada vez son más los datos que están accesibles en la web en el ámbito agropecuario. Sin embargo, el desarrollo de sistemas industriales que saquen partido de ellos necesita que todos estos recursos no estén solamente distribuidos en la web, sino concentrados en un “data hub” que permita analizarlos y proveer un valor añadido. En la historia de la industria agropecuaria, los procesos de registro y almacenamiento de datos han sido manuales. A pesar de la reciente modernización de dicha industria, los procesos de recolección de los datos siguen siendo en la mayoría manuales. Dicha falta de automatización genera que el formato de los datos siga una forma de publicación adaptada a la edición sencilla mediante la creación de archivos Excel. Sin embargo, este formato no se adapta al requerido por los desarrolladores ya que se necesita un formato unívoco y constante de los datos, donde toda la información está concentrada en el mismo lugar y que permita fácil acceso por lenguajes de programación y aplicaciones. Ejemplos de estos sistemas son bases de datos o interfaces web que permite al desarrollador una integración sencilla y natural con sus aplicaciones. Por dichas razones, el primer paso que tiene que dar un desarrollador es pasar de un modelo adaptado al público, por ejemplo, en ficheros de hojas de cálculo, a un modelo adaptado al

programador, en base de datos o servicios web. Sin embargo, si en cada proyecto que se pone en marcha, es necesario volver a implementar el proceso de lectura y adaptación de los datos, el coste del proyecto incrementa notablemente. Es por ello, que la disponibilidad de una infraestructura que aporte una sistemática de trabajo conlleva un claro ahorro de costes y simplificación de trabajos, además de, comodidad, seguridad y adaptabilidad al desarrollador. Para poder trabajar con los datos, es necesario traerlos a una base de datos propia que permita hacer operaciones con ellos sin ninguna dificultad. No tiene sentido que se tenga que ir manualmente a buscar los datos cada vez que se publican y este es el principal motivo para crear dicha infraestructura. De este modo, el principal problema que se aborda es la creación de una infraestructura encargada de la extracción de grandes cantidades de información de manera periódica y eficiente, detectando las nuevas fuentes de información y evitando la re-extracción de datos previos. Los diferentes recursos escogidos son publicados en la web en un determinado instante de tiempo y necesitan ser extraídos en ese momento. De este modo la infraestructura desarrollada se encarga de verificar que los nuevos recursos se encuentran en la web y de extraerlos para almacenarlos en una base de datos.

Otro problema que se aborda es la lectura de ficheros y la identificación de patrones en las fuentes de datos para el desarrollo de propuestas generalistas. Para este problema se han creado diferentes lectores que son capaces extraer la información relevante de los ficheros. Un mismo lector sirve para leer diferentes recursos que cumplen un mismo patrón.

Además, la gran novedad de este trabajo y que no se encuentra en las alternativas existentes en el mercado es la extracción de los recursos de forma periódica. En el mercado actual se encuentra el término Web scraping. Web scraping es el proceso de recopilar información de forma automática de la web. Está muy relacionado con la indexación de la web, la cual indexa la información de la web utilizando un robot y es una técnica universal adoptada por la mayoría de los motores de búsqueda. Sin embargo, el Web scraping se enfoca más en la transformación de datos sin estructura en la web (como el formato HTML) en datos estructurados que pueden ser almacenados y analizados en una base de datos central, en una hoja de cálculo o en alguna otra fuente de almacenamiento. Alguno de los usos del Web scraping son la comparación de precios en tiendas, la monitorización de datos relacionados con el clima de cierta región, la detección de cambios en sitios webs y la integración de datos en sitios webs. Entre las herramientas más usadas para esta tarea están webscraper.io y import.io. La diferencia del proyecto es que estas herramientas no ofrecen la recolección de manera periódica y no permiten trabajar sobre los ficheros publicados, sino que trabajan directamente sobre el código de la página web.

En cuanto a lo personal, la motivación para realizar este proyecto ha sido la posibilidad de realizar una obra útil e interesante, llevando el sello personal y obteniendo el reconocimiento por los demás de esta realización y de los resultados obtenidos. También, hay que destacar que ha sido posible desarrollar competencias personales e incrementar las responsabilidades y la mejora del rendimiento, y a su vez, las posibilidades de promoción como consecuencia del software desarrollado. Se trata de un gran escaparate en el ámbito de en el ámbito de procesos Extracción, Procesado y Carga (ETL) sobre grandes volúmenes de información.

1.3 Contexto Tecnológico

El procesamiento por lotes se trata de la ejecución de un programa sin el control o supervisión directa del usuario de grandes volúmenes de datos. Generalmente, este tipo de ejecución se utiliza en tareas repetitivas sobre grandes conjuntos de información, ya que sería tedioso y propenso a errores realizarlo manualmente. Los programas que ejecutan por lotes suelen especificar su funcionamiento mediante scripts o procedimientos en los que se indica qué se quiere ejecutar y, posiblemente, qué tipo de recursos se necesitan reservar. Hoy en día, existen diferentes entornos para el procesamiento de grandes volúmenes de datos. Spring Batch (Spring, 2020) fue diseñado para manejar el procesamiento por lotes empresarial tradicional en la Java Virtual Machine (Máquina virtual Java, 2020). Fue diseñado para aplicar patrones bien entendidos que son comunes en el procesamiento por lotes de la empresa y hacerlos convenientes en un marco para la JVM. Spark (The Apache Software Foundation, 2020), por otro lado, fue diseñado para casos de uso de big data y machine learning. Esos casos de uso tienen diferentes patrones, desafíos y objetivos que un sistema por lotes empresarial tradicional, y eso se refleja en el diseño del marco. Además, construido y diseñado por el equipo de la Plataforma Hadoop, Marmaray es un marco basado en plug-ins construido sobre el ecosistema Hadoop. Los usuarios pueden agregar soporte para ingerir datos desde cualquier fuente y dispersarse en cualquier sumidero aprovechando el uso de Apache Spark.

En este proyecto, se ha seleccionado el stack de Java mediante el entorno Spring Batch para desarrollar la parte de Backend que extraerá y adaptará los recursos, porque tanto su implementación de patrones de lotes comunes, como el procesamiento y la partición basados en fragmentos, le permite crear aplicaciones de lotes escalables que sean lo suficientemente resistentes para sus procesos más críticos. Además, tiene soporte para archivos, bases de datos relacionales y NoSQL mediante Spring Data y mensajes a través de Apache Kafka y RabbitMQ.

Quartz (Software AG, 2020) se ha utilizado para la programación periódica de la extracción de datos. Se trata de un entorno de programación de tareas periódicas, también llamado "scheduling", de código abierto que provee funcionalidad avanzada para la calendarización de tareas en Java. Con esta funcionalidad, se trata de una excelente opción que nos proporciona un entorno robusto de scheduling, con integración en aplicaciones empresariales que requieran de alta disponibilidad mediante sus características de persistencia y clustering, adicionalmente ofreciendo la posibilidad de integrarlo con cualquier entorno sobre el cual se desarrolle nuestra aplicación.

Desde el punto de vista del desarrollo rápido de aplicaciones, existen varias soluciones existentes en el mercado, entre ellas Spring Boot (Spring, 2020) o el stack MERN (Mongo, Express, React y NodeJS).

Implementado en Java, Spring Boot permite un rápido inicio de una aplicación productiva. La idea de Spring Boot es permitir una fácil y sencilla ejecución, minimizando la cantidad de complicaciones que conlleva la puesta en marcha de una aplicación. Spring Boot implementa un fácil manejo de las dependencias basado en “anotaciones”. Spring Boot incorpora dependencias y configuraciones de Spring básicas que permiten un despliegue simple y rápido. NodeJS, desarrollado principalmente en JavaScript, utiliza un modelo de E/S sin bloqueo controlado por eventos y de un solo thread. Esto lo hace increíblemente eficiente y liviano. Perfecto para aplicaciones muy intensivas en datos que

necesitan operar en tiempo real en equipos distribuidos. Para el desarrollo de una aplicación web completa es necesaria la combinación de productos de software. Una de los más conocidas actualmente es el stack MERN (Mongo, Express, React y NodeJS). MERN es un stack en el que se usa JavaScript tanto en el cliente como en el servidor. Es uno de los entornos de desarrollo de código abierto en más rápido crecimiento. Ayuda a los desarrolladores y equipos a través de herramientas que reducen el tiempo de administración del sistema y permiten el despliegue rápido de aplicaciones web, móvil y API.

Para este proyecto se han utilizado ambas opciones, Spring Boot para el desarrollo de la aplicación web encargada de la planificación de trabajos por lotes, ya que permite la integración con la infraestructura de extracción de recursos implementada con Spring Batch. Y, una combinación de tecnologías stack MERN para el desarrollo de la aplicación web encargada de la presentación de casos de explotación debido a su rápido despliegue.

El despliegue en producción se ha llevado a cabo usando la tecnología de contenedores Docker (Docker Inc, 23) y Docker-Compose. Docker se trata un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux para permitir que contenedores independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales. Docker-Compose básicamente define cada uno de los contenedores que se quiere implementar, además de ciertas características para cada implementación del contenedor. Describe explícitamente cómo quiere implementar la aplicación de varios contenedores en el archivo docker-compose.yml.

En cuanto a herramientas de trabajo y desarrollo, se han utilizado las anteriormente comentadas y algunas como Google Drive para almacenar los diferentes recursos, Skype y Discord como servicios utilizados para la comunicación, seguimiento y videoconferencias, y un tablero de proyectos en Trello para la gestión de tareas. Todas ellas están documentadas en el Anexo B.

1.4 Alcance, objetivos y limitaciones

El objetivo principal es la creación de una infraestructura tecnológica para la extracción de recursos de información disponibles en Internet. El proyecto hace hincapié en la generalización de los procesos de Extracción, Procesado y Carga (ETL) (Extract, transform and load, 2020) de cara a permitir la adición de nuevos recursos según éstos se vayan identificando. Se busca también la identificación de patrones en las fuentes de datos para el desarrollo de diferentes lectores de información que se encargan de extraer los recursos en la web. Dichos recursos son procesados para finalmente ser almacenados en una base de datos. El sistema se valida con la integración de recursos distintos, como datos del porcino en España y Europa, y cereales como arroz, cebada, maíz y trigo. A su vez se ha completado el proyecto con el desarrollo de una aplicación web que muestre posibilidades de explotación.

Se trata de realizar un software automatizado de extracción de datos ya que de este modo se puede ayudar a liberar a los usuarios, dándoles más tiempo para concentrarse en las actividades principales

en lugar de las tareas repetitivas de recopilación de datos. La automatización permite racionalizar todo el proceso desde el momento en que los datos ingresan en la red hasta que se almacenan en un almacén de datos después de ser procesados, eliminando la necesidad de trabajo manual.

El alcance del proyecto se divide en tres partes. Primero, la identificación de patrones en las fuentes de datos para el desarrollo de propuestas generalistas y la creación de una infraestructura para el procesamiento de dichos recursos por medio de trabajos por lotes mediante la herramienta Spring Batch. Además de la planificación de estos trabajos con la biblioteca de programación de tareas de java Quartz Scheduler y el almacenamiento de los datos procesados en una base de datos.

Segundo, la creación de una aplicación web para la planificación de los trabajos de extracción, donde se permita crear, modificar, parar, reiniciar, eliminar y activar inmediatamente un trabajo. Los usuarios podrán visualizar una lista de los trabajos programados y una serie de información como es el nombre del trabajo, la fecha cuando se creó, la última fecha de activación, la próxima fecha en la cual se activará de nuevo y el estado del trabajo.

Por último, la creación de una aplicación web para la presentación de casos de explotación mediante una arquitectura stack MERN. En esta aplicación se van a poder visualizar los diferentes recursos extraídos de una manera mucho más fácil, con el uso de diferentes gráficas y tablas con los precios históricos.

En cuanto a las limitaciones del proyecto, tener en cuenta que los diferentes recursos almacenados en la web se encuentran en ficheros con un formato único. La variación de los formatos de publicación hace que haya que ir monitorizando continuamente los procesos de extracción de datos y hacer modificaciones en la tecnología. Si los responsables de estas fuentes de información cambian o modifican el formato de estos documentos, la infraestructura estaría obligada a alterar los lectores que realizan la extracción conforme al nuevo formato. Esto mismo ha pasado con la extracción de datos del porcino, ya que en el año 2020 los precios de la carne de cerdo pasaron de publicarse de la hoja 10 a la hoja 12.

El sistema se ha construido para que se puedan incorporar nuevos lectores de manera directa. Por un lado, si se quieren extraer recursos cuyo formato sigue el mismo patrón de los lectores ya creados, basta con indicar donde se encuentran la información y qué partes del archivo quieren ser extraídas. Por otro lado, si se quieren incorporar nuevos recursos con un formato diferente sería necesario la creación de nuevos lectores. La creación de este nuevo tipo de lector no implica un problema ya que su implementación es bastante sencilla.

Otra posible limitación sería la desaparición de fuentes de datos, que tiene un problema parecido al de la variación, pero que no lleva la necesidad de reprogramar si no cancelar los automatismos. Un caso de ejemplo encontrado durante el desarrollo de este trabajo es la desaparición de la lonja de Albacete.

Además, la aparición o descubrimiento de nuevas fuentes de información también obligaría a hacer modificaciones en el sistema. Los nuevos recursos se deben incorporar para poder tener el sistema de explotación operativo y competitivo.

1.5 Organización de la memoria

El documento se inicia con una descripción del contexto de trabajo y tecnológico. Después de una breve introducción sobre cuál ha sido la motivación y los diferentes problemas abordados, se describe cuál ha sido el alcance y los objetivos para llevar a cabo en este proyecto, así como un análisis de las posibles limitaciones a los que se deberá hacer frente durante el transcurso del trabajo.

Después, se describe el trabajo desarrollado, donde se da a conocer la pila del producto, mediante una recopilación de las entradas de pila junto con una breve explicación de lo que se ha trabajado. En el apartado posterior se discute la arquitectura que se ha diseñado para el proyecto, el cómo se ha implementado y su despliegue. Una vez explicada la pila del producto, así como los aspectos que conciernen al diseño del software, se pasa a explicar la planificación del trabajo, las pruebas realizadas y demás aspectos técnicos del desarrollo. Y por último se han aportado unas breves conclusiones en las que se explica lo que se ha conseguido en este trabajo y los conocimientos aprendidos.

Finalmente se incluyen diferentes anexos. En el Anexo A, se encuentra una sección de definiciones usadas a lo largo de la memoria. El Anexo B y Anexo C describen las diferentes herramientas utilizados para el desarrollo de este proyecto. En el Anexo D se detalla cómo se ha realizado la búsqueda y descarga de los ficheros almacenados en la Web. El almacén de certificados de Java y cómo se importa un certificado se describe en el Anexo E. Se encuentra en el Anexo F la descripción de las pruebas realizadas en el software. Finalmente, en el Anexo F se muestra el diagrama de módulos del sistema completo y en el Anexo G, un manual de usuario de ambas aplicaciones desarrolladas.

2. TRABAJO DESARROLLADO

2.1 Requisitos del sistema

En esta sección se van a explicar los diferentes requisitos que ofrece el proyecto. Como se ha visto, se pretende crear una infraestructura que posibilite la extracción periódica de recursos agropecuarios y su integración en una base de datos, haciendo hincapié en la generalización de los procesos de Extracción, Procesado y Carga (ETL). Además de, el desarrollo de dos aplicaciones web completamente independientes, una para la gestión de estos trabajos ETL y otra para la presentación de recursos extraídos.

Historias de usuario	
1	El **sistema** podrá extraer, procesar y cargar (ETL) periódicamente recursos de información del ámbito agropecuario almacenados en la web. 1. El sistema deberá poder extraer datos de recursos de información del ámbito agropecuario almacenados en la web. 2. El sistema deberá poder procesar estos datos según sus criterios. 3. El sistema deberá poder cargar los datos procesados en una base de datos.
2	Como **usuario** quiere poder planificar los trabajos ETL: crear, modificar, parar, reiniciar, eliminar y activar inmediatamente un trabajo. 1. Los usuarios deberán poder crear un nuevo trabajo ETL introduciendo un nombre y dos posibilidades de tiempo: ● Introduciendo un fecha concreta. ● Introduciendo una expresión cron. El sistema verificará que dicho nombre no ha sido usado anteriormente y mostrará un mensaje de error. 1. Los usuarios deberán poder modificar el nombre, fecha de activación o la expresión cron de un trabajo. 2. Los usuarios deberán poder parar un trabajo. 3. Los usuarios deberán poder reiniciar un trabajo siempre y cuando esté parado. 4. Los usuarios deberán poder eliminar un trabajo siempre y cuando esté parado.

	5. Los usuarios deberán poder activar un trabajo en ese instante. En todos los casos, el usuario deberá poder visualizar un mensaje de confirmación o error.
3	Como **usuario** quiero poder visualizar los trabajos ETL programados. 1. Los usuarios deberán poder visualizar una lista de los trabajos. 2. Los usuarios deberán poder visualizar el nombre del trabajo, la fecha que se creó, la última fecha de activación, la próxima fecha en la cual se activará de nuevo y el estado del trabajo.
4	Como **usuario** quiero poder visualizar una presentación de los resultados obtenidos mediante los trabajos ETL. 1. Los usuarios deberán poder visualizar en una aplicación web diferentes gráficas de los datos extraídos y almacenados en la base de datos. 2. Los usuarios deberán poder visualizar en una aplicación web los precios históricos de los datos extraídos y almacenados en la base de datos.

2.2 Arquitectura software del sistema

En esta sección se va a hacer una descripción de la arquitectura software de todo el sistema. Primero, se muestra un modelo conceptual de todos los componentes, y después se va a ir explicando uno a uno, detallando de que se encarga cada uno, además de, su despliegue y diseño.

2.2.1 Modelo conceptual

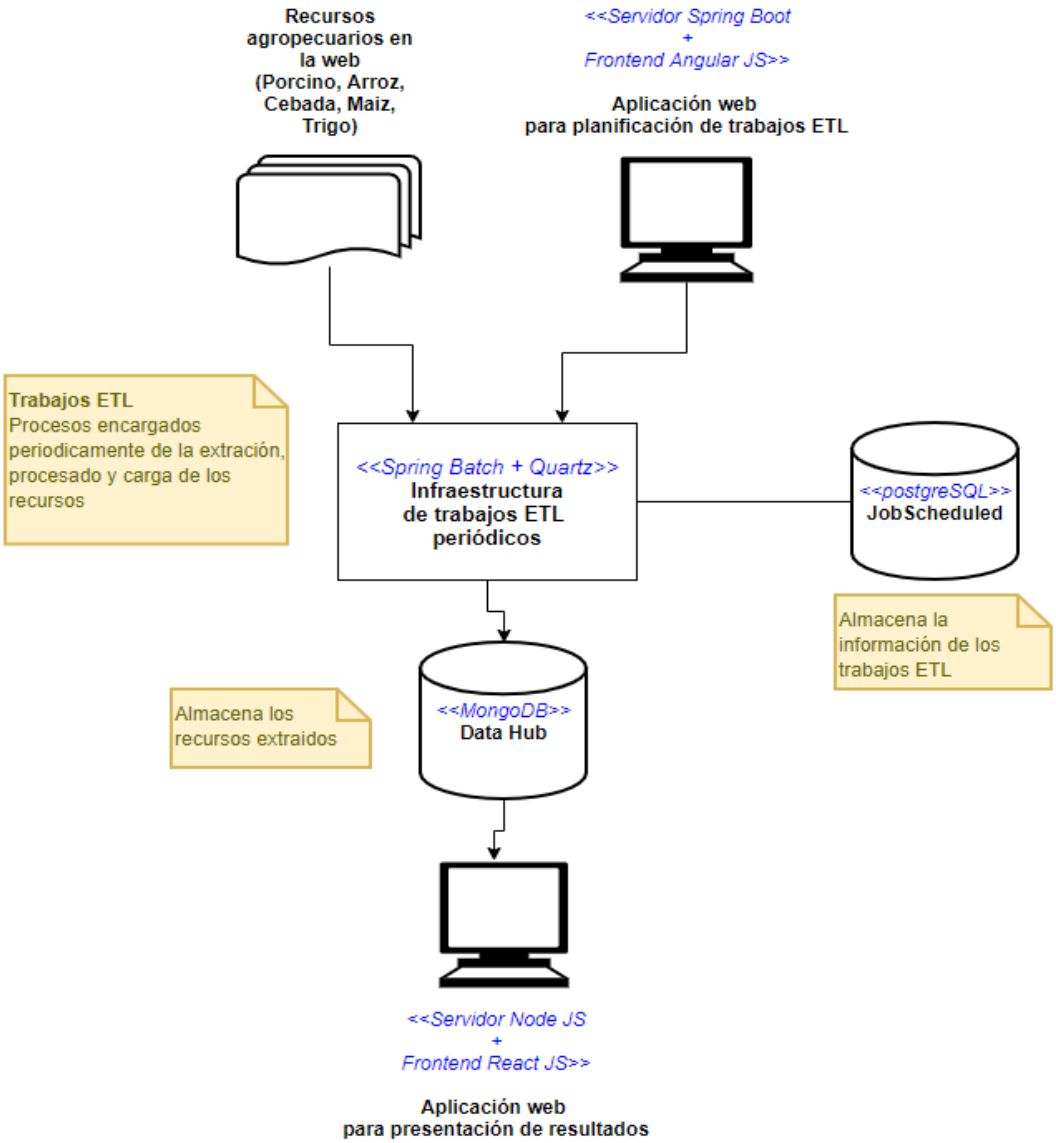

Figura 1: Arquitectura del sistema

Como se muestra en la Figura 1, la arquitectura general del proyecto consta de tres sistemas:

1 Infraestructura de extracción de recursos mediante trabajos ETL periódicos

Infraestructura que posibilita la extracción periódica de recursos agropecuarios disponibles en la red y su integración en una base de datos (Data Hub). Este sistema hace hincapié en la generalización de los procesos de Extracción, Procesado y Carga (ETL) de cara a permitir la adición de nuevos recursos según éstos se vayan identificando. Se realiza el desarrollo de diferentes lectores de información que se encargan de extraer los recursos en la web. Dichos recursos son procesados para finalmente ser almacenados en una base de datos. El sistema desarrollado se ha diseñado con la integración de dos recursos distintos como es el porcino y los cereales.

2 Aplicación web para la planificación de trabajos ETL

Aplicación de interfaz de usuario que permite planificar los trabajos ETL: crear, modificar, parar, reiniciar, eliminar y activar inmediatamente un trabajo. También se pueden visualizar una lista de los trabajos programados.

3 Aplicación web para la presentación de resultados

Aplicación web que permite al usuario visualizar los resultados obtenidos de la extracción de los diferentes recursos. Se ha desarrollado usando una combinación de tecnologías denominada stack MERN.

Para la persistencia de datos, se ha hecho uso de una base de datos Mongo llamada DataHub, en la cual se van a almacenar los diferentes recursos extraídos y otra con la tecnología PostgreSQL para almacenar la información de los trabajos periódicos.

Diagrama de componentes y conectores

Partiendo del modelo conceptual de la Figura 1, se detallan los diversos componentes del proyecto y cómo se produce la comunicación entre ellos. En las siguientes subsecciones se explica cada componente en detalle.

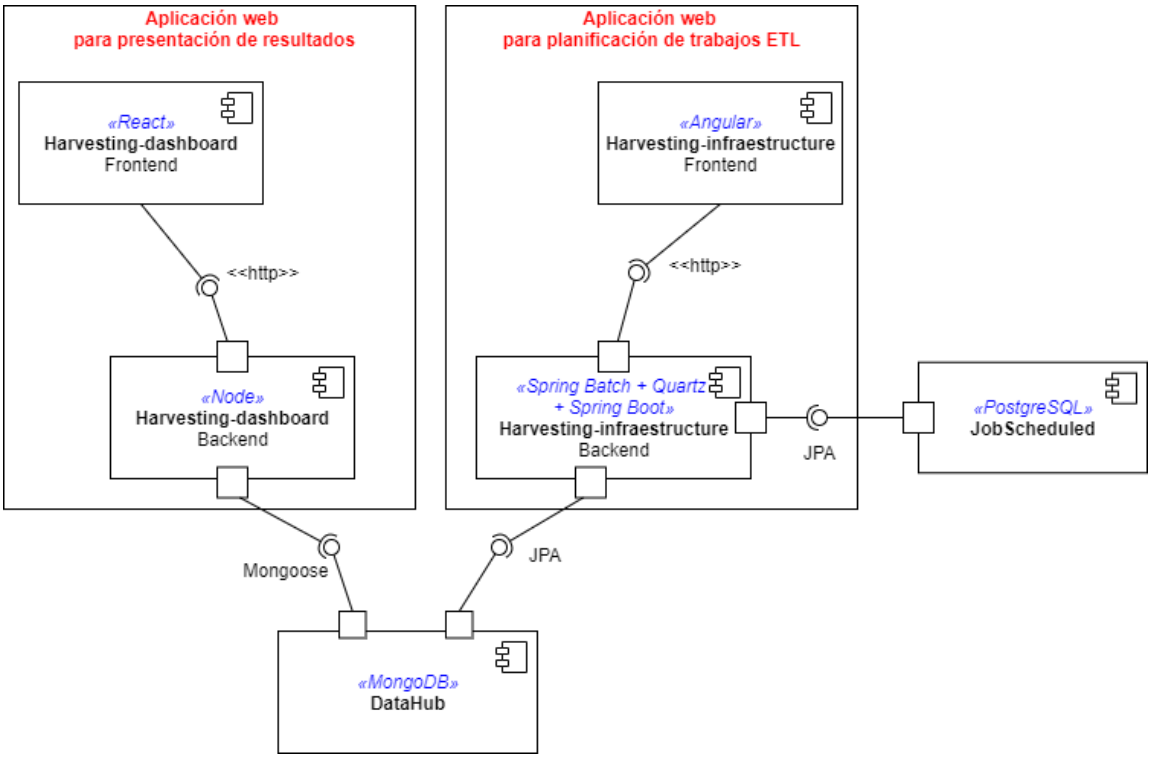

Figura 2: Diagrama de componentes y conectores

2.3 Infraestructura de extracción de recursos mediante trabajos ETL periódicos

Componente encargado de la infraestructura de extracción de recursos periódicamente. Se ha seleccionado el stack de Java mediante el entorno Spring Batch para desarrollar la parte de Backend porque resulta uno de los lenguajes más cómodos y con mayor documentación. Esta herramienta se ha usado para la extracción de los diferentes tipos de datos.

Spring Batch utiliza los objetos denominados Jobs, que son entidades que encapsulan todo un proceso por lotes. A cada trabajo se le asigna un nombre que será el utilizado por la librería Quartz para poder ser localizado y lanzada cuando se produzca el evento programado. A su vez, un Job es simplemente un contenedor para instancias de Step. Un Job combina múltiples pasos que pertenecen lógicamente en un flujo y permite la configuración de propiedades globales para todos los pasos, como la reiniciabilidad. Un Step es un objeto de dominio que encapsula una fase secuencial independiente de un trabajo por lotes. Por lo tanto, cada trabajo se compone completamente de uno o más pasos. A su vez, como se observa en la Figura 3, cada paso tiene exactamente un ItemReader, un ItemProcessor y un ItemWriter, encargados de realizar la extracción, procesamiento y carga respectivamente. En este proyecto, se tiene un trabajo que contiene dos pasos diferentes, uno para la extracción de los datos del porcino y otro para los datos de los cereales.

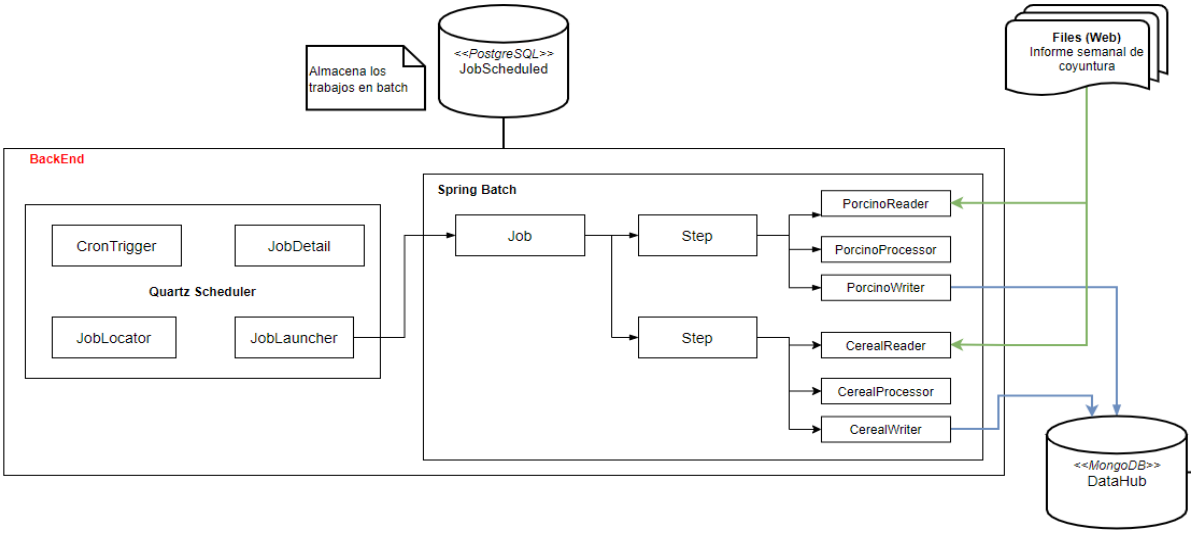

Figura 3: Infraestructura de extracción de recursos detallada

Para la extracción periódica, se ha utilizado el entorno Quartz Scheduler. Con este entorno, cada tarea se especifica mediante un objeto JobDetail y un objeto CronTrigger. JobDetail contienen un nombre de la tarea, un grupo, el nombre del trabajo que debe ser lanzado cuando salte el evento y la clase para que sepa el tipo de trabajo que se ejecutará. Esta clase es la que se encarga de lanzar el trabajo ETL implementado con Spring Batch.

CronTrigger es el encargado de almacenar una expresión cron. Se permite especificar horarios de disparo como "todos los viernes al mediodía" o "todos los días laborables y las 9:30 a.m.", o incluso "cada 5 minutos entre las 9:00 a.m. y las 10:00 a.m. todos los lunes y miércoles y viernes durante enero".

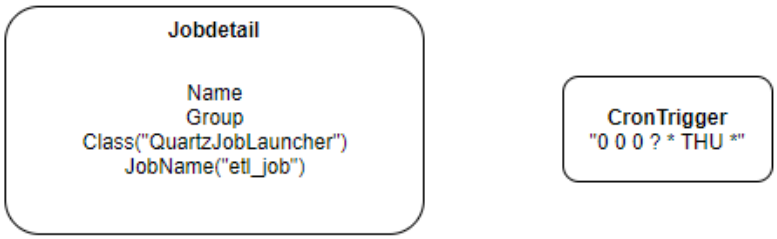

Figura 4: Ejemplo de instancia Quartz

2.3.1 Comunicación Quartz y Spring Batch

Como se describe en la Figura 5, Spring nos proporciona varias formas de planificar las tareas a través de Quartz, por ejemplo, creando una clase que extienda el interfaz QuartzJobBean que a su vez es la encargada de encontrar el trabajo de creado con Spring Batch y lanzarlo para que comience el proceso ETL. El objeto QuartzJobBean actúa como un puente entre Spring Batch y Quartz.

Figura 5: Relación entre Quartz y Spring Batch

Como se observa en la Figura 4, la clase QuartzJobLauncher ha sido la que se ha pasado como parámetro a la instancia de Quartz y esta clase será la que se ejecutará cuando el evento programado por la expresión cron salte.

```
/**
 * The Class QuartzJobLauncher.
 *
 */
public class QuartzJobLauncher extends QuartzJobBean {
```

Figura 6: Clase que extiende el interfaz QuartzJobBean

Cuando se produce un evento desencadenante, se realiza una llamada al método executeInternal de la clase QuartzJobBean. Este método toma los JobDetail asignados a la tarea en Quartz, y mediante el objeto JobLocator busca en ellos el nombre del trabajo previamente almacenado. Una vez que se tiene el trabajo, este se ejecuta mediante el objeto JobLauncher.

```
@Override
protected void executeInternal(JobExecutionContext context) throws JobExecutionException {
    try {
        Job job = jobLocator.getJob(jobName);
        JobExecution jobExecution = jobLauncher.run(job, new JobParameters());
        log.info("{}_{} was completed successfully", job.getName(), jobExecution.getId());
    } catch (Exception e) {
        log.error("Encountered job execution exception!");
    }
}
```

Figura 7: Ejecutar un Job en la clase QuartzJobBean

Una vez lanzado el trabajo de Spring Batch, se ejecutan los diferentes pasos en orden secuencial. El primer paso se encarga de extraer los datos del porcino y el segundo paso de los datos del cereal. Toda esta información queda almacenada en la base de datos llamada DataHub.

Cada paso comienza con la fase de extracción y lectura de los datos y realiza las siguientes acciones:

- Extraer los datos desde los sistemas de origen (Web).
- Analizar los datos extraídos para verificar que los datos extraídos cumplen la pauta o estructura que se esperaba. Si no fuese así, los datos serán rechazados.
- Convertir los datos a un formato preparado para iniciar el proceso de transformación.

2.3.2 Descarga de ficheros

Antes de iniciarse los diversos pasos de cada trabajo, se accede a la url de la web donde se encuentran los ficheros que se desean obtener. Para la extracción de los recursos, se ha utilizado la técnica denominada Web scraping, proceso de recopilar información de forma automática de la web obteniendo su código HTML. Se tiene que verificar el código de respuesta HTTP del servidor para asegurar de que la URL está disponible (estado HTTP 200) y entonces se produce la obtención del código HTML del sitio web. En este código, almacenado en una cadena de caracteres, se ha realizado una búsqueda mediante expresiones regulares de los ficheros que se desean descargar. Una vez obtenido el nombre de los ficheros, son descargados uno a uno comprobando que no haya sido descargado antes, es decir, que no se encuentre en el almacenamiento local de la máquina. Los nombres de fichero descargados son enviados a cada lector para que se produzca la extracción de los datos.

2.3.3 Lectura de ficheros Excel

Una vez se ha descargado los ficheros, el lector realiza la extracción de los datos. Se han implementado diferentes lectores debido a la incompatibilidad de los patrones en los ficheros Excel. En este proyecto es necesario la lectura de archivos Excel, lo cual no se corresponde con ninguna de las opciones que proporcionan los lectores de Spring Batch. Es por eso por lo que se han tenido que crear lectores personalizados para la lectura de estos ficheros. Para ello se ha hecho uso de Apache POI (The Apache Software Foundation, 2020), biblioteca para el tratamiento de archivos de Excel y Word desde programas Java.

Primero de todo, se ha tenido que hacer una comprobación a la hora de abrir los ficheros debido a que la librería utilizada trata de forma diferente si es un fichero tiene extensión .xls o .xlsx.

PRODUCTO	MERCADO REPRESENTATIVO	Semana 19 04-10/05 2020	Semana 20 11-17/05 2020	Variación €
Trigo Blando Panificable	Albacete	197,00	195,00	**-2,00**
	Ávila	195,00	192,00	**-3,00**
	Barcelona	208,00	206,00	**-2,00**
	Burgos	188,20	186,60	**-1,60**
	Cádiz	207,00	207,00	**0,00**

Figura 8: Formato de los datos correspondientes al Trigo

PRODUCTO	MERCADO REPRESENTATIVO	Semana 19 04-10/05 2020	Semana 20 11-17/05 2020	Variación €
Cebada Pienso	Albacete	157,80	157,40	**-0,40**
	Ávila	164,00	162,00	**-2,00**
	Burgos	161,00	160,00	**-1,00**
	Cádiz	170,00	170,00	**0,00**
	Ciudad Real	162,50	160,70	**-1,80**

Figura 9: Formato de los datos correspondientes a la Cebada

Se han creado lectores personalizados de una manera parametrizada, en los cuales se le asigna la hoja donde se encuentran los datos, la ruta de los ficheros a procesar, fila inicial, fila final, tipo de recurso y las columnas donde se encuentran los diferentes datos. Se ha hecho de esta manera con la intención de que un mismo lector pueda servir para diferentes fuentes que cumplen el mismo patrón de lectura. Como se muestra en la Figura 8 y en la Figura 9, el formato de ambos ficheros es idéntico y para ello se utiliza el mismo lector, pero con diferentes parámetros.

Para los datos del Porcino se ha tenido que crear otro lector complemente diferente ya que el formato de este fichero, como se muestra en la Figura 10va por columnas y se lee de otra manera.

Cada lector va extrayendo la información de los ficheros dato por dato y los almacena en la base de datos.

MERCADO REPRESENTATIVO	SELECTO (nivel menor de grasa)			NORMAL (nivel normal de grasa)		
	Semana 19 04-10/05 2020	Semana 20 11-17/05 2020	Variación €	Semana 19 04-10/05 2020	Semana 20 11-17/05 2020	Variación €
Barcelona	--	--	--	1,35	1,32	**-0,03**
Huesca	1,39	1,30	**-0,09**	1,37	1,28	**-0,09**

Figura 10: Formato de los datos correspondientes al Porcino

Cada lector no termina de extraer datos hasta que se devuelve el valor null, es decir, se han acabado de leer todos los ficheros descargados. Cada dato leído es transformado mediante los objetos ItemProcessor. La fase de transformación de los procesos de ETL aplica una serie de reglas de negocio o funciones sobre los datos extraídos para convertirlos en datos que serán cargados. En esta transformación se modifican los caracteres erróneos y se construye la fecha con un objeto Date debido a que el lector extrae datos de tipo carácter. De este modo, se podrán hacer búsquedas y ordenaciones por fecha.

Por último, los datos procedentes de la fase anterior (fase de transformación) son cargados en el sistema de destino. Los datos extraídos y procesados son enviados al objeto ItemWriter que se encarga de almacenarlos en la base de datos. Si un dato está repetido no será almacenado.

A todo esto, se le añade una herramienta de registro de logs que ayudan a detectar y analizar los errores y problemas en tiempo de ejecución.

2.3.4 Validación de datos

A la hora de extraer los datos de los ficheros, es posible encontrarse son datos erróneos o con datos no existentes. Para ello, se ha realizado una validación de datos en el momento de la lectura. Si el lector se encuentra datos erróneos como los observados en la Figura 11, antes de enviarlos al objeto ItemProcessor se realiza una validación del dato leído para ver si se trata de un dato de tipo numérico o si la celta está en blanco. En caso de ser un dato no válido, el lector continúa con la siguiente línea.

MERCADO REPRESENTATIVO	SELECTO			
	Semana 52 23-29/12 2019	Semana 01 30/12-05/01 2020	Variación €	Se 2
Barcelona	--	--	--	

Figura 11: Formato erróneo de los datos correspondientes al Porcino

Persistencia

La base de datos con tecnología PostgreSQL llamada JobScheduled se encarga de almacenar de forma persistente la información de los diferentes trabajos de extracción periódicos. La API que permite la comunicación con este componente es el lenguaje SQL, consta de operaciones para insertar (insert), modificar (update) y eliminar (delete) cualquier tabla de la base de datos.

Para trabajar desde este entorno con la base de datos de nuestra aplicación se ha empleado además JPA (Java Persistence API), ofrecido por Java para implementar un entorno Object Relational Mapping (ORM) que permite interactuar con la base de datos por medio de objetos. Para más concreción, la implementación utilizada ha sido Hibernate, permite trabajar directamente con las entidades.

2.4 Aplicación web para la planificación de trabajos ETL

En esta sección se detalla el componente correspondiente con la aplicación web que gestiona la planificación de los trabajos programados. Permite planificar los trabajos ETL: crear, modificar, parar, reiniciar, eliminar y activar inmediatamente un trabajo. También se pueden visualizar una lista de los trabajos programados. Se hace una diferenciación entre la interfaz de usuario y el servidor.

2.4.1 Frontend

Componente encargado de las vistas de interfaz de usuarios de aplicación de planificación de trabajos que renderiza los ficheros de visualización en el navegador web. Dichas vistas hacen uso de diferentes servicios que ofrece el servidor Harvesting-dashboard Backend, como se observa en la Figura 2, mediante llamadas HTTP a una API REST.

El desarrollo de la parte de interfaz de usuario se ha realizado con el entorno Angular, ya que permiten un desarrollo rápido, pero con una gran experiencia de usuario.

Como se observa en la Figura 12, en la parte superior se permite crear un nuevo trabajo introduciendo un nombre y seleccionando el tipo de recurso que se quiere extraer (Porcino o Cereal). Para la ejecución temporal de este trabajo se permite introducir una fecha concreta o una expresión cron.

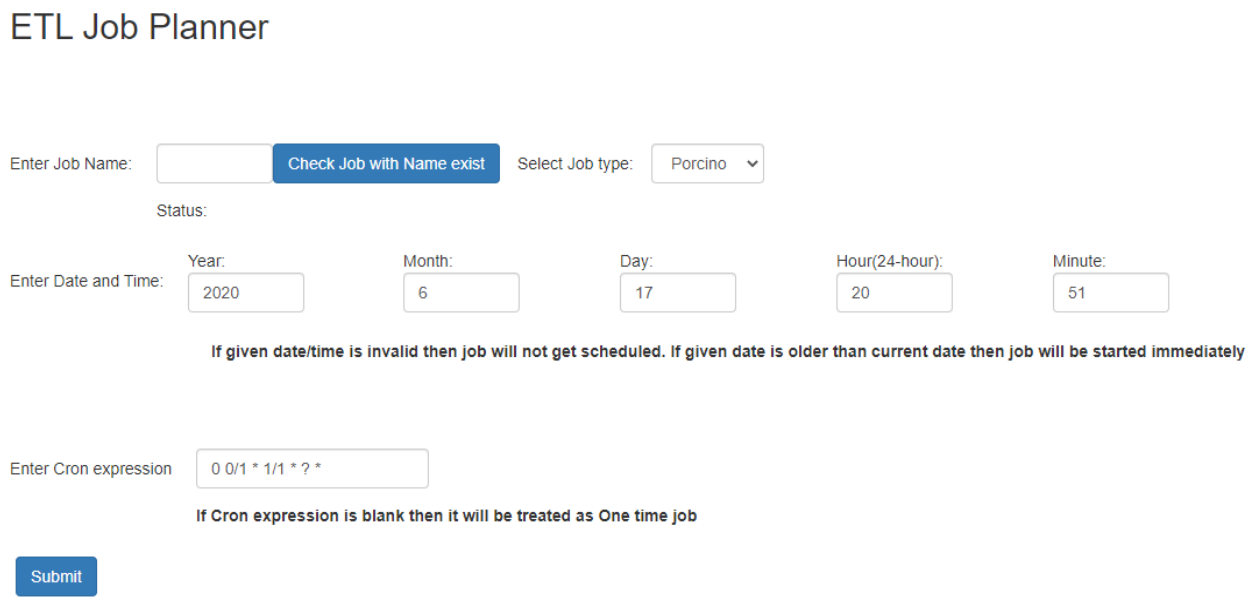

Figura 12: Creación de un trabajo

Figura 13: Listado de trabajos programados

En la parte inferior, como se detalla en la Figura 13, se visualiza una lista de todos los trabajos programados. Se muestra el nombre, fecha de creación, última fecha de ejecución, próxima fecha de ejecución, diferentes acciones y el estado actual del trabajo. Cada botón sirve para ejecutar una de estas acciones: Ejecutar en este instante, parar, reiniciar, eliminar, parar ejecución en marcha o editar información. Cada funcionalidad realiza una llamada HTTP a la API REST de servicios (API REST Planificación de trabajos, s.f.) que ofrece el Backend de esta aplicación.

Estados de los trabajos ETL

La Tabla 1 describe los posibles estados que puede tener un trabajo en Quartz a lo largo del proceso.

Tabla 1: Estados de un trabajo

ESTADO	Comportamiento
SCHEDULED	Trabajo programado para ser ejecutado
RUNNING	Trabajo corriendo en ese instante
PAUSED	Trabajo parado

COMPLETED	Trabajo finalizado con éxito
ERROR	Trabajo finalizado con error
BLOCKED	Trabajo bloqueado

2.4.2 Backend

Componente encargado de gestionar los servicios de planificación de trabajos en el lado del servidor. Tiene como puerto una API REST mediante llamadas HTTP.

En la implementación del Backend se ha utilizado el entorno de trabajo Spring Boot el cual se encarga del desarrollo de aplicaciones y contenedor de inversión de control, y de código abierto para la plataforma Java. Se ha escogido esta herramienta frente a otras debido a la gran cantidad de servicios que posee, y una gran cantidad de buenas características, entre las cuales se han tenido en cuenta:

- Cuenta con plantillas para diversas tecnologías como JPA la cual es utilizada en este proyecto.
- Data access Framework: Permite a los desarrolladores usar 'persistence' APIs como JDBC. Para almacenar información en bases de datos, lo cual era especialmente importante en nuestro proyecto.
- Spring MVC Framework: Permite construir aplicaciones web basadas en la arquitectura MVC.

En el enfoque de Spring Boot para construir sitios web, las solicitudes HTTP son manejadas por un controlador. Se ha construido un controlador mediante la anotación @RestController para manejar las solicitudes HTTP y gestionar los diferentes servicios que ofrece la aplicación como son listar, crear, modificar, parar, reiniciar, eliminar y activar inmediatamente un trabajo. Cada una de estas funciones hace una llamada a las operaciones que ofrece la librería Quartz para la gestión de tareas.

En la Tabla 1, se han detallado los diferentes servicios que ofrece la aplicación mediante las siguientes llamadas HTTP. De este modo una API consigue que los desarrolladores interactúen con los datos de la aplicación de un modo planificado y ordenado. Permite la comunicación entre el cliente web y el servidor.

Tabla 2: Servicios de la aplicación de planificación de trabajos

Llamada API REST	*Comportamiento*
/scheduler/checkJobName/{jobName}	Chequea si existe el nombre del trabajo
/scheduler/delete/{jobName}	Elimina el trabajo
/scheduler/isJobRunning/{jobName}	Comprueba si el trabajo está en marcha
/scheduler/Jobs	Devuelve todos los trabajos programados
/scheduler/jobState/{jobName}	Devuelve el estado del trabajo
/scheduler/pause/{jobName}	Para el trabajo
/scheduler/resume/{jobName}	Reinicia el trabajo previamente parado
/scheduler/schedule/{jobName}	Crea un nuevo trabajo en Quartz
/scheduler/start/{jobName}	Pone en marcha el trabajo
/scheduler/stop/{jobName}	Para el trabajo en marcha
/scheduler/update/{jobName}	Modifica los datos del trabajo

2.5 Aplicación web para la presentación de resultados

En esta sección se detalla el componente correspondiente con la aplicación web que se encarga de la presentación de los resultados extraídos. Se hace una diferenciación entre la interfaz de usuario y el servidor.

2.5.1 Frontend

Componente encargado de las vistas de interfaz de usuarios de la aplicación de presentación de resultados que renderiza los ficheros de visualización en el navegador web en forma de vistas de interfaz de usuarios. Dichas vistas hacen uso de diferentes servicios que acceden al servidor Harvesting-dashboard Backend mediante llamadas HTTP a una API REST.

Para el desarrollo completo de la aplicación web ha sido necesaria la combinación de distintos productos de software, la conocida actualmente como stack MERN (Mongo, Express, React y NodeJS). MERN es un stack en el que se usa JavaScript tanto en el cliente como en el servidor. Es uno de los entornos de desarrollo de código abierto en más rápido crecimiento. Permite el despliegue rápido de aplicaciones web, móvil y API. React ha sido el entorno encargado de implementar la interfaz de usuario.

Figura 14: Pantalla principal web presentación de resultados

En la Figura 14 se muestra como es la pantalla principal de la aplicación, en la cual encontramos diferentes componentes.

Situada en la izquierda de la pantalla principal como se observa en la Figura 15, se encuentra una barra lateral desde la cual es posible seleccionar entre los distintos recursos agropecuarios que tiene la aplicación para visualizar su información.

Figura 15: Panel de categorías

A su vez, existe un componente formado por dos botones que se encuentra en la parte superior derecha. El botón con la etiqueta Europa es para acceder a los precios del porcino en los mercados europeos y el botón con la etiqueta Histórico de precios es para visualizar el histórico de precios de ese recurso en España.

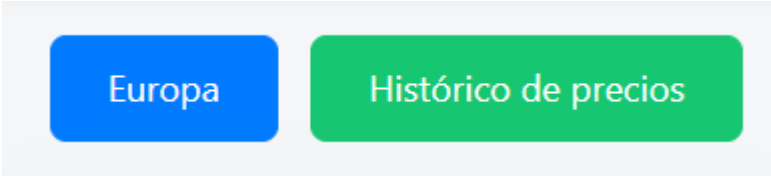

Figura 16: Botones de navegación

En la parte superior de la pantalla principal, como se ve en el componente de la Figura 17, es posible visualizar el precio de la última semana y la variación con respecto a la anterior y una gráfica temporal de los precios de las últimas cuatro semanas. Además, se produce una diferenciando según el tipo del recurso.

Figura 17: Cuadros de resumen

En la parte inferior de la pantalla principal, se encuentra una gráfica secuencial donde se muestra el precio en los diversos mercados a lo largo del tiempo, como se puede observar en la Figura 18. La gráfica permite acotar rangos entre dos fechas y agregar o eliminar la visualización de los mercados.

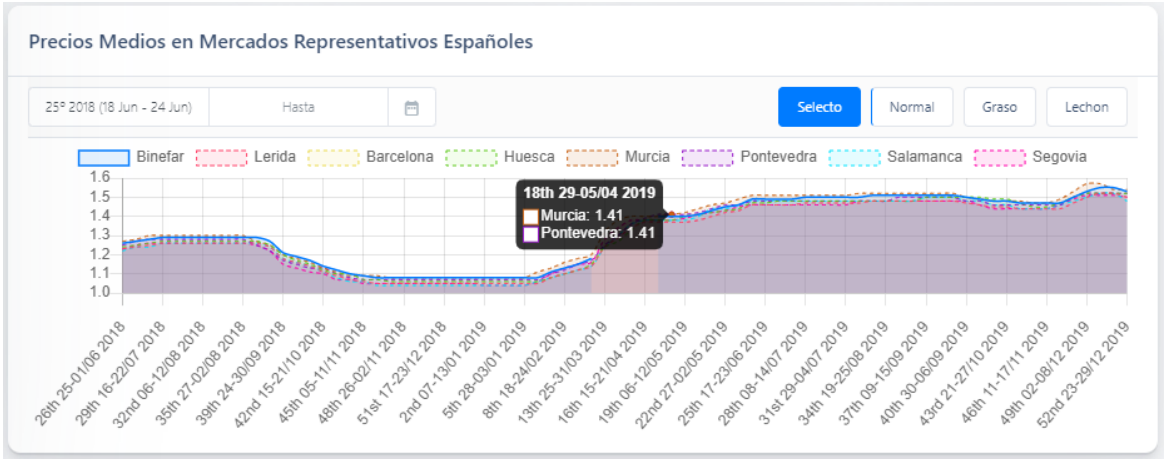

Figura 18: Gráfica temporal de precios y mercados

Pulsando en el botón *Europa,* el cual se observa en la Figura 16, aparece una nueva pantalla con una gráfica de los datos del porcino correspondientes a mercados europeos. En la gráfica se permite seleccionar el tipo, acotar rangos entre dos fechas y agregar o eliminar la visualización de los mercados.

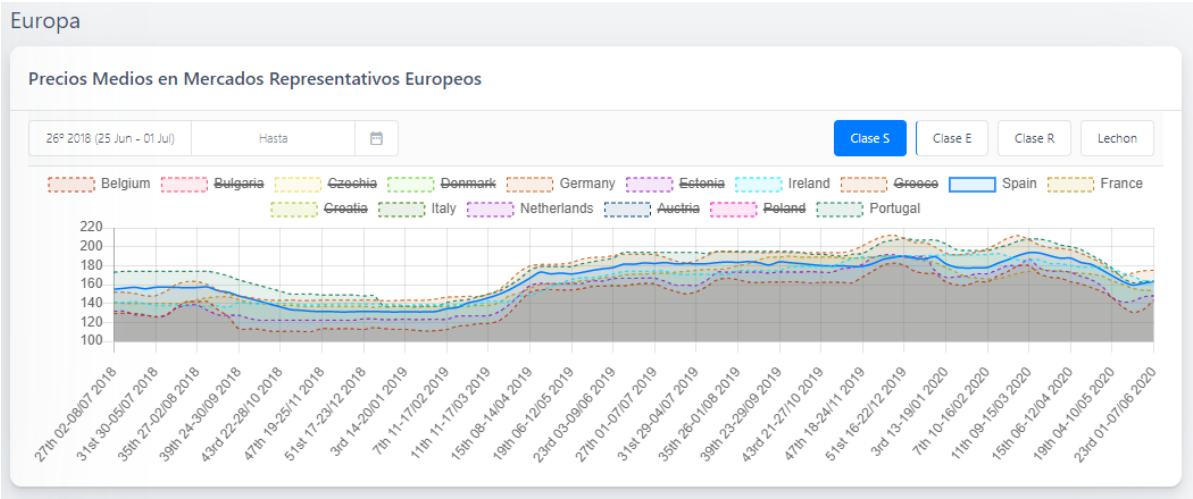

Figura 19: Gráfica del Porcino de mercados europeos

Pulsando en el botón de *Histórico de precios*, en los botones de navegación descritos en la Figura 16, aparece una nueva pantalla con las diferentes tablas para cada tipo donde se muestran los mercados y su precio en cada semana. Se permite la búsqueda y ordenar por fecha, precio de mayor a menor o viceversa.

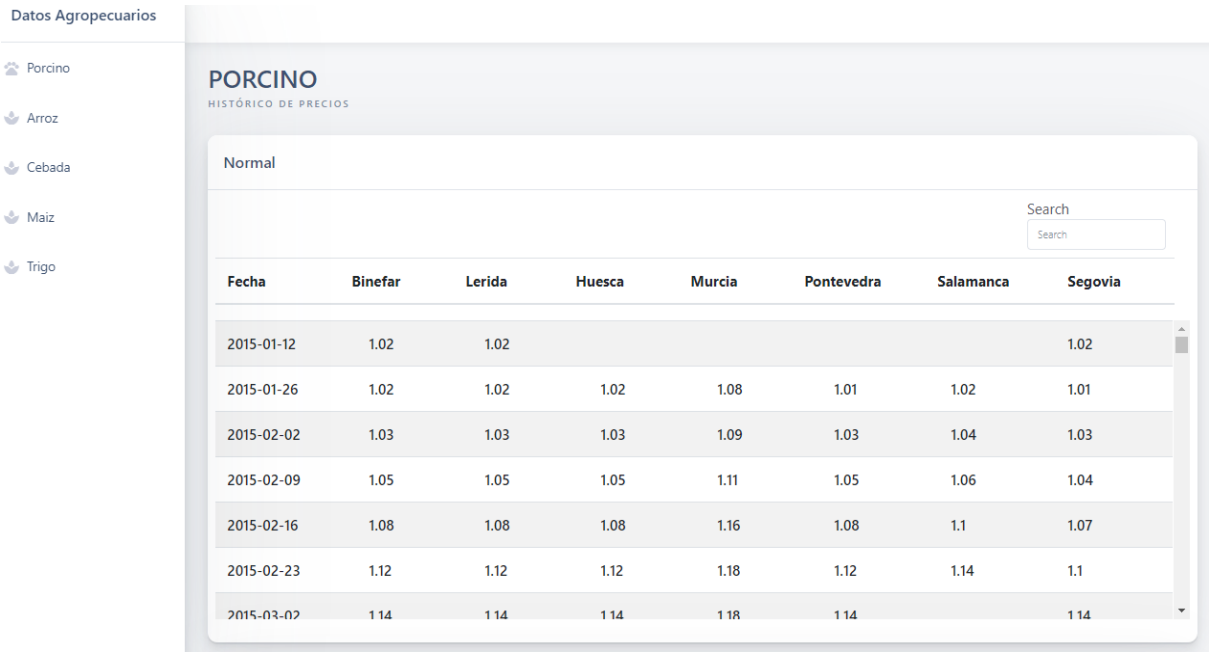

Figura 20: Histórico de precios del Porcino

Seleccionando otro dato del panel de selección de categorías, se cambia a una pantalla distinta con la información de este nuevo recurso. La Figura 21 muestra la pantalla de información del arroz.

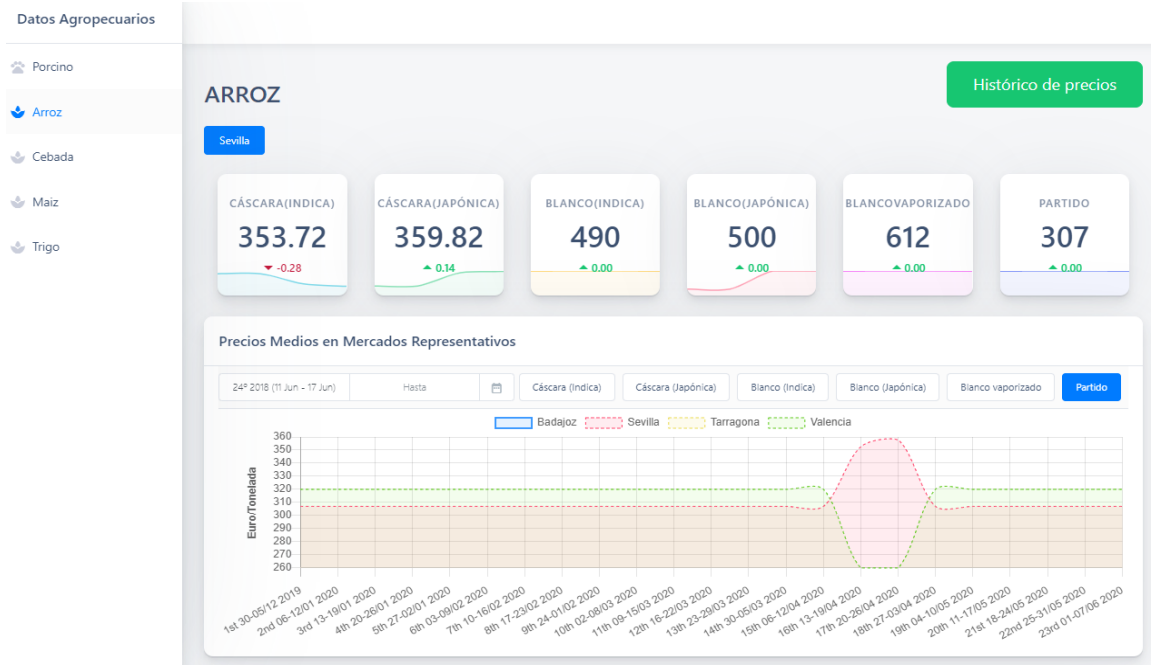

Figura 21: Pantalla de información del Arroz

2.5.2 Backend

Componente que implementa el lado del servidor la aplicación de presentación de datos. Tiene como puerto una API REST mediante llamadas HTTP. Node JSse ha usado para la implementación del lado

del servidor y mediante el módulo Express que ofrece esta misma tecnología, se han creado diversas rutas para el acceso desde el exterior y mediante controladores se obtiene la información de la base de datos. Para obtener esta información, se ha usado otro módulo de NodeJS llamado Mongoose (Mongoose, s.f.), el cual permite acceder a la base de datos de una manera muy simple. Este módulo ofrece validación de datos a la hora de acceder a los recursos mediante la definición de esquemas de datos. De este modo, un recurso con datos erróneos no podrá ser almacenado.

Se han detallado los diferentes servicios que ofrece la aplicación mediante las siguientes llamadas HTTP. De este modo una API consigue que los desarrolladores interactúen con los datos de la aplicación de un modo planificado y ordenado. Permite la comunicación entre el cliente web y el servidor. Los datos se encuentran en formato JSON.

Tabla 3: Servicios de la aplicación de presentación de resultados

Llamada API REST	***Comportamiento***
/historical/aggregate/{pork}/{market}/{type}/{date}	Devuelve los datos del porcino. Se permite la búsqueda por nombre, mercado, tipo y fechas.
/historical/europe/{pork}/{market}/{type}/{date}	Devuelve los datos del porcino en Europa. Se permite la búsqueda por nombre, mercado, tipo y fechas.
/cereal/aggregate/{pork}/{market}/{type}/{date}	Devuelve los datos del cereal. Se permite la búsqueda por nombre, mercado, tipo y fechas.
/cereal/markets/{pork}/{type}	Devuelve los mercados del cereal. Se permite la búsqueda por nombre, mercado, tipo y fechas.

2.6 Data Hub

Componente encargado de almacenar de forma persistente toda la información correspondiente a los diferentes recursos extraídos de la web. Desde la aplicación de presentaciones de resultados se ha utilizado el conector llamado Mongoose (Mongoose, s.f.), el cual permite acceder de una manera muy simple.

En esta base de datos, se encuentran las colecciones para almacenar los recursos extraídos. Se tienen tres colecciones diferentes: Cereal, EuropePork y Historical. Cada colección almacena los documentos de un dato distinto, cereales, porcino en Europa y porcino en España respectivamente. Además, para cada colección se ha creado un índice por medio del identificador de cada documento para mejorar la velocidad de las operaciones, permitiendo un rápido acceso a los documentos de la colección.

En las siguientes subsecciones se describe el modelo de datos utilizado para almacenar los diferentes recursos extraídos.

2.6.1 Animal

- *Date*: se compone de la fecha estándar y un timestamp del momento de la extracción.
- *Animal*: especie de animal.
- *Type*: tipo de animal.
- *Market*: mercado correspondiente del animal.
- *Value*: valor en euros del animal.

Animal

{JSON}

```
{
    date :
    {
          standard  : Date,
          timestamp : Date
    },
    animal : String,
    type   : String,
    market : String,
    value  : Number
}
```

```
{
    "date" : {
        "standard" : "2015-07-20T00:00:00Z",
        "timestamp" : "2015-07-20T08:00:00Z"
    },
    "animal" : "pork",
    "type" : "Selecto",
    "market" : "Binefar",
    "value" : 1.3
}
```

Figura 22: Formato JSON y ejemplo del recurso Animal

2.6.2 EuropePork

- *Date*: se compone de la fecha estándar y un timestamp del momento de la extracción.
- *Animal*: especie de animal.
- *Type*: tipo de animal.
- *Markets*: lista de mercados. Un mercado se compone de país y precio.

EuropePork

{JSON}

```
{
    date :
    {
          standard  : Date,
          timestamp : Date
    },
    animal  : String,
    type    : String,
    markets : [
        {
              price: Number,
              country: String
        }
        ]
}
```

```
{
    "date" : {
        "standard" : "2015-07-20T00:00:00Z",
        "timestamp" : "2015-07-20T08:00:00Z"
    },
    "animal" : "pork",
    "type" : "ClassS",
    "markets" : [{
        "price" : 124,
        "country" : "Belgium"
    }]
}
```

Figura 23: Formato JSON y ejemplo del recurso EuropePork

2.6.3 Cereal

- *Date:* se compone de la fecha estándar y un timestamp del momento de la extracción.
- *Cereal:* nombre del cereal.
- *Type:* tipo de cereal.

- *Market:* mercado del cereal.
- *Value:* valor en euros del cereal.

Cereal

```
{                                   {JSON}
    date :
    {
          standard  : Date,
          timestamp : Date
    },
    cereal : String,
    type   : String,
    market : String,
    value  : Number
}
```

```
{
    "date" : {
        "standard" : "2015-07-20T00:00:00Z",
        "timestamp" : "2015-07-20T08:00:00Z"
    },
    "cereal" : "TRIGO",
    "type" : "Duro",
    "market" : "Zaragoza",
    "value" : 260
}
```

Figura 24: Formato JSON y ejemplo del recurso Cereal

2.7 Despliegue

Como se ve en la Figura 25, el paso a producción se ha llevado a cabo mediante el despliegue del sistema en cuatro contenedores de Docker en una máquina Linux que proporciona una IP pública. Un primer contenedor llamado DataHub almacena la base de datos Mongo que recoge los diferentes recursos agropecuarios. Un segundo contenedor, JobScheduled donde se encuentra la base de datos PostgreSQL encargada de recoger la información de los trabajos de extracción. En un tercer contenedor con nombre Harvesting-infraestructure se despliega la infraestructura encargada de la extracción y procesado de los recursos, además de la aplicación de planificación de trabajos. Y un último contenedor denominado Harvesting-dashboard se ejecuta toda la aplicación de presentación de resultados.

La construcción automática del software y despliegue de todo el sistema se ha realizado mediante docker-compose. Esta herramienta básicamente define cada uno de los contenedores que se quiere implementar, además de ciertas características para cada implementación del contenedor.

Se han definido dos ficheros Dockerfile para que el comportamiento pueda ser reproducido en cualquier lugar. Uno de ellos construye el contenedor que almacena la aplicación web de presentación de resultados y otro la infraestructura encargada de la extracción de datos y la aplicación de planificación de trabajos. Una vez se tienen estos ficheros, se han definido los servicios que componen el proyecto en un archivo docker-compose.yml para que se puedan ejecutar juntos en un entorno aislado. En él, se han documentado los cuatro contenedores que forman sistema: un contenedor PostgreSQL para la base de datos que almacena la información de los trabajos periódicos, un contenedor MongoDB para almacenar los diferentes recursos extraídos y los dos contenedores anteriormente comentados.

Una vez que se tiene este archivo de descripción de la implementación de varios contenedores, se puede desplegar el sistema completo en una sola acción organizada por el comando de la consola docker-compose up.

Para el control de versiones del proyecto se ha usado la herramienta Git.

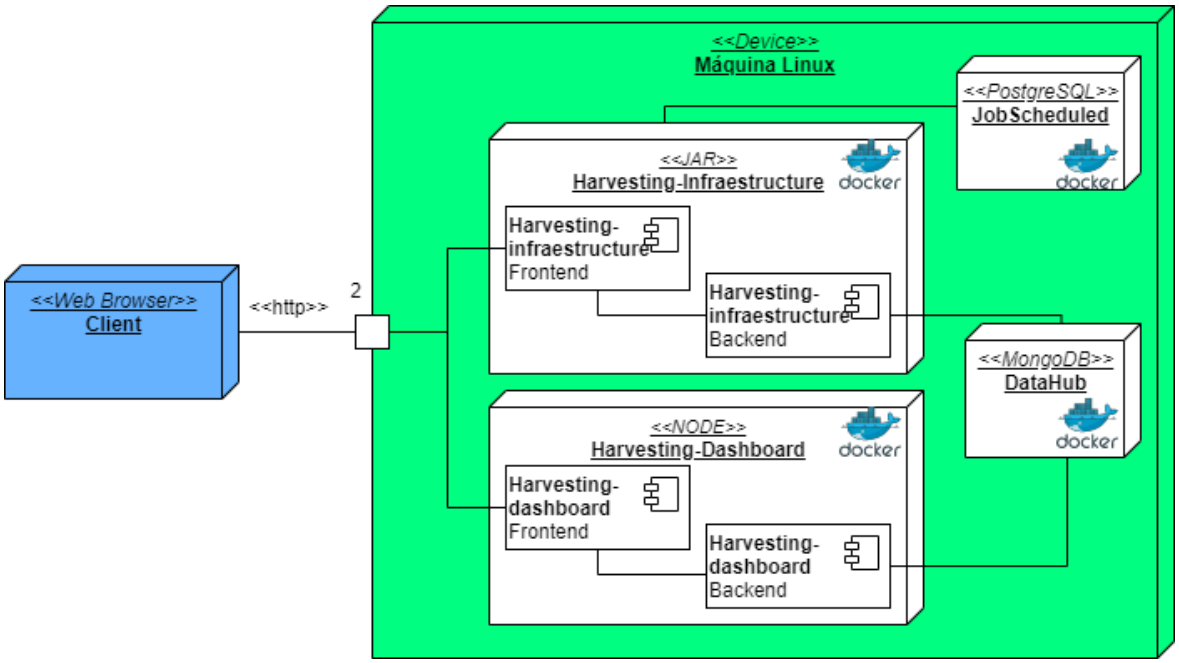

Figura 25: Diagrama de despliegue

2.8 Pruebas

Para la realización de pruebas en el Backend se ha decidido utilizar Junit (JUnit, s.f.), ya que ofrece mucha versatilidad para realizar pruebas con Spring Boot. Se han realizado test unitarios y se ha utilizado la herramienta JaCoCo para determinar la cobertura.

Se han realizado diferentes pruebas unitarias para verificar el correcto funcionamiento de los sistemas.

Búsqueda de fichero almacenado en la web

Se comprueba el funcionamiento de la técnica Web scraping descargando el código HTML de una página Web donde se encuentra el fichero que se desea descargar y mediante expresiones regulares se hace la búsqueda de ese nombre de fichero. Si el nombre encontrado es igual al nombre del fichero la comprobación es correcta.

Descarga de fichero almacenado en la web

Se comprueba la correcta descarga del fichero almacenado en la web usando la función que realiza dicha acción. Se le pasa como parámetro la dirección del fichero publicado en la web y tras realizar

la descarga se comprueba que el nombre del archivo descargado en el almacenamiento de la maquina es equivalente al nombre que se espera.

Extracción y procesamiento de los datos

Para la comprobación de que los datos que se extraen de un fichero son válidos, se ha realizado una prueba que valida la lectura de estos datos.

PRODUCTO	MERCADO REPRESENTATIVO	Semana 17 20-26/04 2020	Semana 18 27/04-03/05 2020	Variación €
Trigo Blando Panificable	Albacete	198,00	197,00	**-1,00**

Figura 26: Datos del Trigo

Esta prueba realiza la lectura de un fichero donde se encuentran los datos del trigo blando panificable de Albacete como se muestra en la Figura 26. Tras hacerse la lectura y procesamiento de la primera línea, se comprueba que los datos extraídos son los datos realizando una comprobación como se describe en la Figura 27.

```
assertEquals(cereal.getCereal(),"TRIGO");
assertEquals(cereal.getType(),"BlandoPanificable");
assertEquals(cereal.getMarket(), "Albacete");
assertEquals(cereal.getValue(), 197, 0);
```

Figura 27: Comprobación de datos

De este modo se comprueba la correcta lectura y procesamiento de los datos.

Para comprobar las aplicaciones web desarrolladas, se han realizado pruebas de estudios con usuarios externos para la evaluación final de la interfaz de usuario.

Se ha empleado la herramienta pgAdmin para gestionar la base de datos PostgreSQL y verificar el correcto almacenamiento de los diferentes trabajos programados.

Otra herramienta utilizada para poner a prueba la API de servicios de las dos aplicaciones web ha sido Postman, esta permite generar el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente. De este modo se han comprobado las respuestas que generan las llamadas a la API.

2.9 Planificación y esfuerzo

Al principio del proyecto se hizo una planificación inicial dividida en diferentes fases.

- Fase 1 (Marzo): estudio de fuentes de datos, y de tecnologías (fundamentalmente Spring Batch, Quartz Scheduler y Mongo).

- Fase 2 (Marzo-Mayo): diseño y desarrollo de la infraestructura general para el procesamiento de trabajos en batch.
- Fase 3 (Abril): diseño y desarrollo de la aplicación web para la planificación de trabajos.
- Fase 4 (Mayo): diseño y desarrollo de la aplicación web para explotación de resultados.
- Fase 5 (Junio): pruebas de integración y despliegue en sistema de producción.

Como se observa en la Figura 28, los plazos establecidos al comienzo se han cumplido correctamente. Además, se ha realizado una reunión semanal durante los meses que ha durado el desarrollo del proyecto para llevar un seguimiento del proyecto.

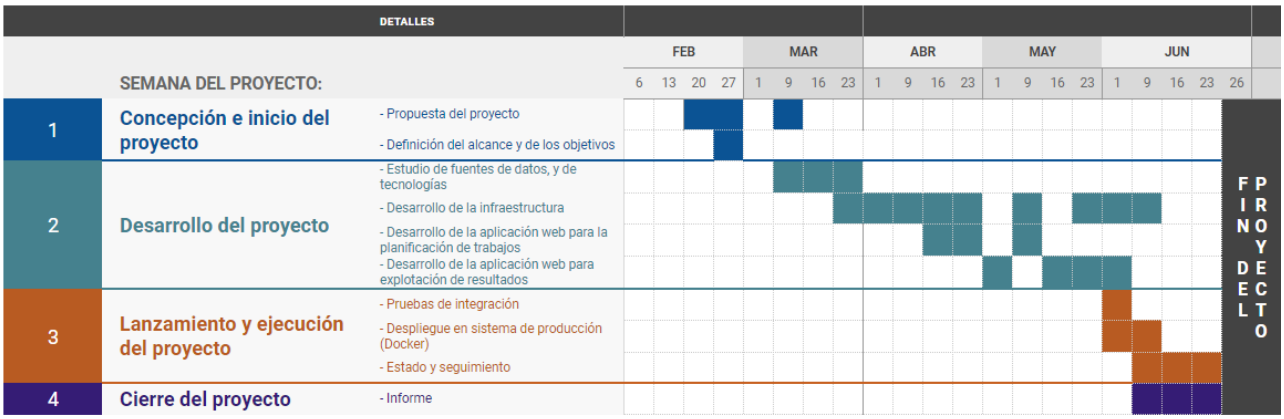

Figura 28: Diagrama de Gantt

En la Tabla 4 se detallan las horas totales llevadas a cabo en el desarrollo del proyecto y las líneas de código que tiene todo el proyecto completo.

Tabla 4: Horas totales y líneas de código del proyecto

Horas totales	216
Líneas de código	200.059

A continuación, se describe el desglose del trabajo realizado según el tipo de actividad como se observa en la Figura 29. Se puede ver que la parte que ha llevado más tiempo ha sido la implementación de la infraestructura encargada de la extracción de los recursos, seguida del desarrollo de las dos aplicaciones web del proyecto. La formación y estudio de las diferentes tecnologías y la gestión del proyecto incluyendo reuniones de seguimiento semanales y el desarrollo del informe han ocupado prácticamente el mismo reparto de tiempo.

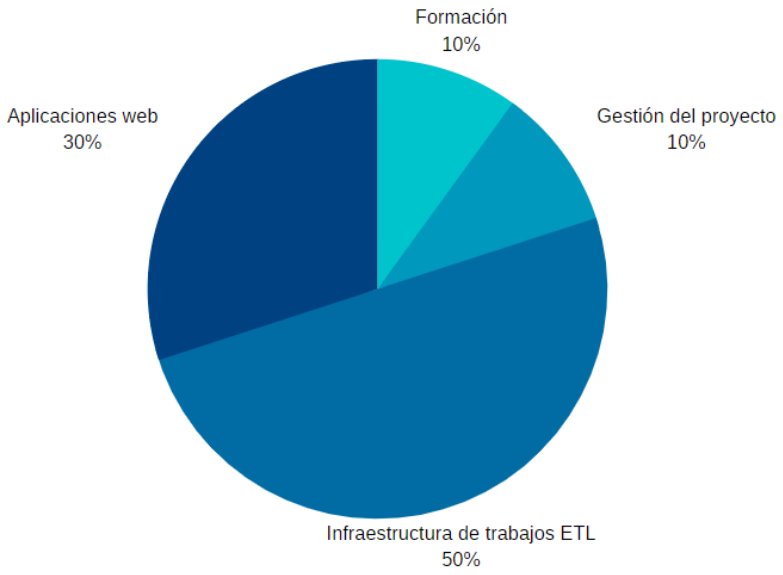

Figura 29: Desglose del trabajo

2.10 Problemas encontrados

La biblioteca de planificación de tareas de java Quartz ofrece una serie de servicios para la gestión de estas tareas, pero no ofrece una interfaz de usuario que permita la fácil utilización de los servicios de forma externa al código. Por ello se ha hecho uso de una solución en forma de aplicación que ofrece un usuario externo (Jayesh, 2017).

La infraestructura realiza la extracción de los diferentes recursos publicados en la web y es por ello por lo que para la descarga de los ficheros se realiza mediante una url donde se encuentran estos recursos. Es necesaria la búsqueda de los diferentes ficheros descargando el código html de la url usada. Este problema se ha resuelto con la técnica denominada Web scraping, detallando el procesa en el Anexo D.

Uno de los problemas encontrado ha sido el cambio de formato en los ficheros. La variación de los formatos de publicación obliga a ir monitoreando continuamente los procesos de extracción de datos y hacer modificaciones en los componentes o directamente añadiendo nuevos lectores. Esto mismo ha pasado con la extracción de datos del porcino, ya que en el año 2020 los precios de la carne de cerdo pasaron de publicarse de la hoja 10 a la hoja 12. El problema se resuelve modificando el lector correspondiente para la extracción de los datos que han cambiado su formato, ajustándose al nuevo formato.

La desaparición de fuentes de datos tiene un problema parecido al de la variación, pero que no lleva la necesidad de reprogramar sino de cancelar los automatismos. Un ejemplo de este problema ha sido la desaparición de la lonja de Albacete, el cual no supone ningún problema ya que los lectores están programados para ser resistentes ante este tipo de imprevistos y el lector ya no extrae estos

datos. Si por algún motivo se dejasen de publicar los recursos escogidos para la extracción, mediante la aplicación construida de planificación de trabajos se podría cancelar dicho proceso.

Otro problema que ha surgido ha sido la lectura de los ficheros donde se encuentran los datos ya que se trata de ficheros Excel. El entorno Spring Batch no ofrece facilidad para la lectura de este tipo de ficheros por lo que se ha tenido que hacer uso de la biblioteca de Java Apache POI mediante lectores personalizados.

Otro problema ocurrido durante el transcurso de la implementación del trabajo ha sido el error del sistema a la hora de realizar la descarga de alguno de los ficheros publicados en la web. Para solucionarlo se ha tenido que importar un certificado autofirmado al almacén de Java para que la aplicación que intenta establecer una conexión SSL confíe en este certificado y permita descargar los ficheros. Se detalla en el Anexo E almacén de certificados de Java.

3. LECCIONES APRENDIDAS Y CONCLUSIONES

3.1 Conclusiones

Hoy en día, conlleva una gran dificultad el hecho de que se tenga que ir manualmente a buscar los datos cada vez que se publican y se quiera trabajar con ellos. Para poder trabajar y operar con datos es necesario extraerlos a una base de datos propia, y por ese motivo se ha creado esta infraestructura que lo permite por medio de automatismos. La aplicación de la automatización aporta ciertas ventajas al usuario como la comodidad, la seguridad y la adaptabilidad y además se traduce en un ahorro económico y temporal. Tener acceso a datos oportunos es imprescindible para tomar mejores decisiones y realizar operaciones comerciales sin problemas.

En cuanto a lo personal, hay que destacar que ha sido posible desarrollar competencias personales e incrementar las responsabilidades y la mejora del rendimiento. A su vez, se han aumentado las posibilidades de promoción como consecuencia del software desarrollado con un sello propio. Se trata de una gran experiencia en el ámbito de tareas repetitivas sobre grandes volúmenes de información.

3.2 Conocimientos adquiridos

Este proyecto comenzó con el estudio de nuevas tecnologías como Spring Batch, Quartz Scheduler y bases de datos Mongo. Por un lado, Spring Batch ha servido para tener gran experiencia en procesamiento de grandes volúmenes de datos. Con Quartz, he aprendido cómo gestionar la calendarización de diferentes tareas en el lenguaje Java.

Anteriormente, solo había trabajado con bases de datos relacionales y con este proyecto, el uso de Mongo me ha ayudado a coger gran soltura con bases de datos no relacionales. Este tipo de almacenamientos son mucho más abiertas y flexibles y permiten adaptarse a necesidades de proyectos mucho más fácilmente que los modelos de Entidad Relación.

En cuanto al desarrollo de aplicaciones, este cuatrimestre había trabajado con la combinación de tecnologías denominada stack MERN y se trata de una buena solución para desarrollar la aplicación de presentación de resultado ya que permite una fácil implementación y despliegue.

Para la lectura de ficheros en Java, se ha aprendido a utilizar la librería Apache POI, la cual permite el tratamiento de archivos de Excel y Word desde programas Java.

3.3 Trabajo futuro

Actualmente, cualquier usuario puede obtener la información de los diferentes recursos almacenada en la base de datos. Por este motivo, sería conveniente añadir algún método de control de usuarios para saber en todo momento quien accede y hace uso de los datos.

Debido a que en la actualidad se está produciendo constantemente la publicación de datos abiertos y, además, dentro del ámbito agropecuario hay multitud de recursos de información que se publican periódicamente en la web, esta infraestructura permite la adición de nuevos recursos según éstos se vayan identificando. La aparición o descubrimiento de nuevas fuentes de información se deben incorporar para poder tener el sistema de explotación operativo y competitivo. Desde el punto de vista de la usabilidad para el usuario, ahora es necesario añadir los lectores que realizan la extracción de los recursos de manera manual en el código. Se podría implementar una solución que facilite la incorporación de lectores construyendo una interfaz de usuario que permita hacer esto de manera automática.

Entre los otros posibles usos, y en una primera vía de aprovechamiento de este TFG, los diferentes recursos de información que se obtienen con esta infraestructura serán utilizados para la integración de técnicas de Inteligencia Artificial para el establecimiento del precio de referencia de la carne de porcino. Tener acceso a datos oportunos es imprescindible para tomar mejores decisiones y realizar operaciones comerciales sin problemas, debido a que muchas empresas dependen de la extracción de datos por lotes. Esto significa que la información disponible para el análisis podría no reflejar los datos operativos más recientes o que las decisiones comerciales cruciales deben basarse en datos históricos. Por lo tanto, una herramienta eficaz de extracción de datos debería permitir la extracción en tiempo real con la ayuda de flujos de trabajo automatizados para preparar los datos más rápido para la inteligencia empresarial.

4. BIBLIOGRAFÍA

API REST Planificación de trabajos. (s.f.). Obtenido de http://51.210.10.251:7080/swagger-ui.html

Beltran. (19 de 11 de 2015). *Importar un Certificado de una CA en el cacerts de la Jdk*. Obtenido de Lo que me interesa de la red: https://loquemeinteresadelared.wordpress.com/2015/11/19/importar-un-certificado-de-una-ca-en-el-cacerts-de-la-jdk/

Commission, E. (23 de 06 de 2020). *Pigmeat statistics*. Obtenido de European Commision: https://ec.europa.eu/info/food-farming-fisheries/farming/facts-and-figures/markets/overviews/market-observatories/meat/pigmeat-statistics_en

Docker Inc. (2020 de 06 de 23). *Docker*. Obtenido de Docker: https://www.docker.com/

experto.dev. (s.f.). Obtenido de https://experto.dev/wp-content/uploads/2015/11/spring-batch-domain-language.png

Extract, transform and load. (13 de 02 de 2020). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Extract,_transform_and_load

Gobierno de España. (23 de 06 de 2020). *Informe Semanal de Coyuntura*. Obtenido de Ministerio de Agricultura, Pesca y Alimentación: https://www.mapa.gob.es/es/estadistica/temas/publicaciones/informe-semanal-coyuntura/default.aspx

IAAA. (23 de 06 de 2020). *Advanced Information Systems Laboratory*. Obtenido de http://iaaa.unizar.es

Jayesh. (31 de 10 de 2017). *Quartz Scheduler + Spring Boot Example*. Obtenido de JavaByPatel: https://javabypatel.blogspot.com/2017/10/quartz-scheduler-spring-boot-example.html

JUnit. (s.f.). Obtenido de https://junit.org/junit5/

Máquina virtual Java. (20 de 03 de 2020). Obtenido de WIkipedia: https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java

Minh, N. H. (18 de 07 de 2019). *Java HttpURLConnection to download file from an HTTP URL*. Obtenido de CodeJava: https://www.codejava.net/java-se/networking/use-httpurlconnection-to-download-file-from-an-http-url

Mongoose. (s.f.). Obtenido de https://mongoosejs.com/

OpenJS Foundation. (23 de 06 de 2020). *NodeJS*. Obtenido de NodeJS: https://nodejs.org/es/

Software AG. (23 de 06 de 2020). *Quartz Scheduler*. Obtenido de http://www.quartz-scheduler.org/

Spring. (11 de 06 de 2020). *Spring Batch*. Obtenido de Reference Documentation: https://docs.spring.io/spring-batch/docs/current/reference/html/index.html

Spring. (12 de 06 de 2020). *Spring Boot 2.3.1*. Obtenido de https://spring.io/projects/spring-boot

The Apache Software Foundation. (03 de 09 de 2020). *Apache POI*. Obtenido de https://poi.apache.org/

The Apache Software Foundation. (26 de 06 de 2020). *Apache Spark*. Obtenido de https://spark.apache.org/

Wikipedia. (10 de 04 de 2020). Obtenido de Datos abiertos: https://es.wikipedia.org/wiki/Datos_abiertos

Anexo A Definiciones

Backend: parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web.

Frontend: parte de un sitio web que interactúa con los usuarios, por eso decimos que está del lado del cliente.

Trabajo ETL*:* Extracción, procesamiento y carga en una base de datos de un recurso de información.

Fecha*:* fecha en un formato AAAA-MM-DD-HH-MM.

Expresión cron*:* es una cadena de texto compuesta por 6 o 7 campos separados por un espacio en blanco que se utiliza para representar instantes o periodos de tiempo. Su principal uso recae en la planificación de ejecuciones de procesos o rutinas programadas. Los campos pueden contener cualquiera de los valores permitidos, junto con varias combinaciones de los caracteres especiales permitidos para ese campo. Los campos son los siguientes:

Expresiones cron pueden ser tan simples como esto: * * * * ? * *

o más complejas, como: 0/5 14,18,3-39,52 * ? ENE,MAR,SEP LUN-VIE 2002-2020

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of week	YES	1-7 or SUN-SAT	, - * ? / L #
Year	NO	empty, 1970-2099	, - * /

Figura 30: Formato de expresión cron

Estado de un trabajo*:* Se trata del estado que tiene el trabajo en cada instante: SCHEDULED, RUNNING, PAUSED, COMPLETED, ERROR, BLOCKED.

Scheduling: trabajo o la actividad de planificar los tiempos en los que se harán determinadas tareas o eventos.

Clustering: área de agrupar un conjunto de objetos de tal manera que los objetos en el mismo grupo (llamado agrupamiento) sean más similares (en cierto sentido) entre sí que con los de otros grupos (agrupamientos). Es una tarea principal de la minería de datos exploratorios, y una técnica común para el análisis estadístico.

Cluster: se aplica a los sistemas distribuidos de granjas de computadoras unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen un único servidor.

Archivo XML: traducido como "Lenguaje de Marcado Extensible", es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

HTTP: Hypertext Transfer Protocol, es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML.

API: abreviatura de Application Programming Interface, es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

REST: abreviatura de Representational State Transfer, es un estilo de arquitectura para diseñar aplicaciones en red. Una API podría considerarse REST si su arquitectura se ajusta a ciertas reglas o restricciones.

HTML: abreviatura de HyperText Markup Language, es el lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet.

JSON: abreviatura de JavaScript Object Notation, es un formato de texto sencillo para el intercambio de datos.

Webpack: es una herramienta de agregación de recursos para aplicaciones web que permite generar una distribución única a partir de un conjunto establecido de assets.

Url: abreviatura de Uniform Resource Locator, es la dirección específica que se asigna a cada uno de los recursos disponibles en la red con la finalidad de que estos puedan ser localizados o identificados.

Anexo B Herramientas de programación

B.1 Spring Batch

Se conoce como sistema por lotes (en inglés batch processing), o modo batch, a la ejecución de un programa sin el control o supervisión directa del usuario (que se denomina procesamiento interactivo). Este tipo de programas se caracterizan porque su ejecución no precisa ningún tipo de interacción con el usuario. Generalmente, este tipo de ejecución se utiliza en tareas repetitivas sobre grandes conjuntos de información, ya que sería tedioso y propenso a errores realizarlo manualmente.

Spring Batch provee funcionalidades esenciales para el procesamiento de grandes volúmenes de registros, logs, transacciones, procesamiento de estadísticas, omisiones, reinicio de procesos, etc. Los procesos por lotes, o batch processing requieren normalmente grandes cantidades de recursos. Este tipo de entorno se utiliza en tareas repetitivas sobre grandes conjuntos de información, ya que es tedioso y propenso a errores realizarlo manualmente. La capacidad de Spring Batch para procesar eficientemente grandes cantidades de datos lo hace ideal para muchos casos de uso. La implementación de los patrones de procesamiento estándar de la industria le permite crear trabajos por lotes robustos en la JVM. Tienen soporte para extraer recursos de archivos, bases de datos relacionales y NoSQL mediante Spring Data y mensajes a través de Apache Kafka y RabbitMQ. Utilizado para implementar la infraestructura que posibilita la extracción periódica de los recursos.

Spring Batch es el estándar para el procesamiento por lotes en la JVM. Su implementación de patrones de lotes comunes, como el procesamiento y la partición basados en fragmentos, le permite crear aplicaciones de lotes escalables de alto rendimiento que sean lo suficientemente resistentes para sus procesos más críticos. Spring Boot proporciona un nivel adicional de características de grado de producción para permitirle acelerar el desarrollo de sus procesos por lotes.

Las clases ItemReader y ItemWriter tienen soporte para archivos, bases de datos relacionales y NoSQL mediante Spring Data y mensajes a través de Apache Kafka y RabbitMQ. Spring Batch tiene la capacidad de manejar la mayoría de los casos de uso.

Un trabajo tiene uno o muchos pasos, cada uno de los cuales tiene exactamente un ItemReader, un ItemProcessor y un ItemWriter.

Se debe iniciar un trabajo con JobLauncher y se deben almacenar en JobRepository los metadatos sobre el proceso actualmente en ejecución.

Job

Entidad que encapsula todo un proceso por lotes. Como es común con otros proyectos de Spring, a Job está configurado mediante un archivo XML o una configuración basada en Java. Esta configuración puede denominarse "configuración de trabajo". Sin embargo, Job es solo la parte superior de una jerarquía general, como se muestra en el siguiente diagrama:

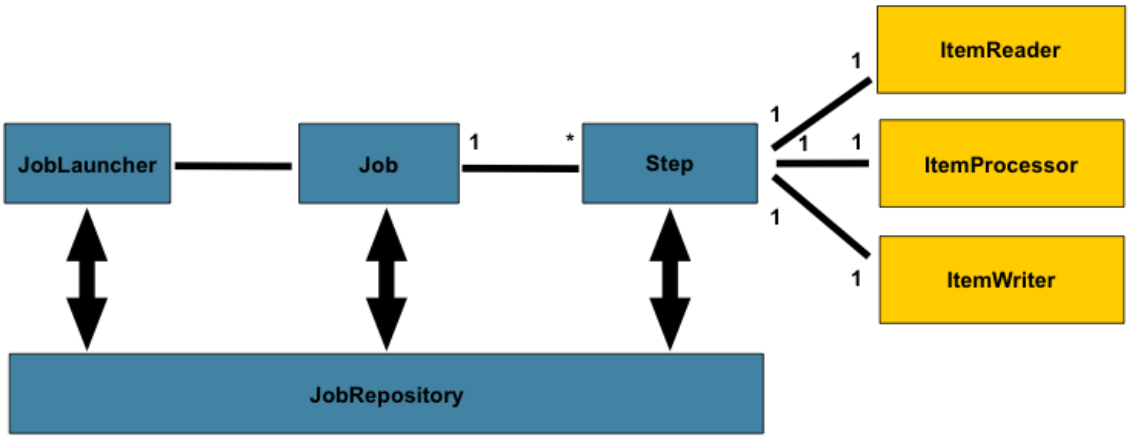

Figura 31: Esquema de Spring Batch (experto.dev, s.f.)

Job es simplemente un contenedor para instancias de Step. Combina múltiples pasos que pertenecen lógicamente en un flujo y permite la configuración de propiedades globales para todos los pasos, como la reiniciabilidad. La configuración del trabajo contiene:

- El simple nombre del trabajo.
- Definición y ordenamiento de Step instancias.
- Si el trabajo es reiniciable o no.

JobInstance

Se refiere al concepto de una ejecución de trabajo lógica. Usar el mismo JobInstance determina si se usa o no el 'estado' de ejecuciones anteriores. Usando un nuevo JobInstance significa "comenzar desde el principio", y el uso de una instancia existente generalmente significa comenzar desde donde lo dejó. Un objeto JobParameters contiene un conjunto de parámetros utilizados para iniciar un trabajo por lotes.

El contrato se puede definir como: JobInstance= Job + identificación JobParameters

JobExecution

Se refiere al concepto técnico de un solo intento de ejecutar un trabajo. Una ejecución puede terminar en fallo o éxito, pero la JobInstance correspondiente a una ejecución dada no se considera completa a menos que la ejecución se complete con éxito.

Un Job define qué es un trabajo y cómo debe ejecutarse, y JobInstance es un objeto puramente organizativo para agrupar ejecuciones, principalmente para permitir la semántica de reinicio correcta. JobExecution, sin embargo, es el mecanismo de almacenamiento primario de lo que realmente sucedió durante una ejecución y contiene muchas más propiedades que deben controlarse y persistir.

Step

Objeto de dominio que encapsula una fase secuencial independiente de un trabajo por lotes. Por lo tanto, cada trabajo se compone completamente de uno o más pasos.

ExecutionContext

Representa una colección de pares clave/valor que el marco persiste y controla para permitir a los desarrolladores un lugar para almacenar el estado persistente que se define en un objeto StepExecution o un objeto JobExecution.

El mejor ejemplo de uso es facilitar el reinicio. Hacerlo le permite al objeto ItemReader almacenar su estado en caso de que ocurra un error fatal durante la ejecución o incluso si se corta la energía. Todo lo que se necesita es poner el número actual de líneas leídas en el contexto.

JobRepository

Mecanismo de persistencia para todos los estereotipos mencionados anteriormente. Proporciona operaciones CRUD para JobLauncher, Job y Step. Cuando Job se inicia por primera vez, JobExecution se obtiene del repositorio y, durante el curso de la ejecución, StepExecution y las implementaciones de JobExecution persisten pasándolas al repositorio. Cuando se utiliza la configuración de Java, la anotación @EnableBatchProcessing proporciona JobRepositor y uno de los componentes configurados automáticamente de fábrica.

JobLauncher

Representa una interfaz simple para iniciar un Job con un conjunto dado de JobParameters.

ItemReader

Abstracción que representa la recuperación de entrada para un elemento Step a la vez. Cuando ItemReader ha agotado los elementos que puede proporcionar, lo indica al regresar null.

Aunque es un concepto simple, un ItemReader es el medio para proporcionar datos de muchos tipos diferentes de entrada. Los ejemplos más generales incluyen:

1. Archivo plano: los lectores de elementos de archivo plano leen líneas de datos de un archivo plano que generalmente describe registros con campos de datos definidos por posiciones fijas en el archivo o delimitados por algún carácter especial (como una coma).
2. XML: proceso XML independientemente de las tecnologías utilizadas para analizar, mapear y validar objetos. Los datos de entrada permiten la validación de un archivo XML contra un esquema XSD.
3. Base de datos: se accede a un recurso de base de datos para devolver conjuntos de resultados que pueden asignarse a objetos para su procesamiento. Las implementaciones de

SQL predeterminadas invocan a RowMapper para devolver objetos, realizar un seguimiento de la fila actual si es necesario reiniciar, almacenar estadísticas básicas y proporcionar algunas mejoras de transacciones que se explican más adelante.

ItemProcessor

Abstracción que representa el procesamiento comercial de un artículo. Mientras ItemReader lee un elemento y los ItemWriter escribe, ItemProcessor proporciona un punto de acceso para transformar o aplicar otro procesamiento comercial. Si, mientras procesa el artículo, se determina que el artículo no es válido, la devolución null indica que el artículo no debe escribirse.

ItemWriter

Abstracción que representa la salida de un Step, permitiendo la escritura de un lote o una porción de elementos a la vez. En general, un ItemWriter no tiene conocimiento de la entrada que debe recibir a continuación y solo conoce el elemento que se pasó en su invocación actual.

Configurar un trabajo

LISTENER (Interceptar ejecución de trabajo)

Durante el curso de la ejecución de un Trabajo, puede ser útil recibir notificaciones de varios eventos en su ciclo de vida para que se pueda ejecutar un código personalizado.

@BeforeJob: función que se ejecuta antes de cada trabajo. Para este proyecto, antes de cada trabajo se descargan los ficheros que van a ser procesados.

@AfterJob: función que se ejecuta después de cada trabajo.

Configuración de Java

Simplemente con la anotación @EnableBatchProcessing, se proporciona una configuración básica para crear trabajos por lotes. Dentro de esta configuración básica, se crea una instancia de StepScope, además de una serie de beans disponibles para ser auto conectados:

- JobRepository - nombre del bean "jobRepository"
- JobLauncher - nombre del bean "jobLauncher"
- JobRegistry - nombre del bean "jobRegistry"
- PlatformTransactionManager - nombre del bean "transactionManager"
- JobBuilderFactory - nombre de bean "jobBuilders"
- StepBuilderFactory - nombre del bean "stepBuilders"

Procesamiento orientado a fragmentos

El procesamiento orientado a fragmentos se refiere a leer los datos de uno en uno y crear 'fragmentos' que se escriben dentro de un límite de transacción. Un elemento se lee de un

ItemReader, se entrega a un ItemProcessor y se agrega. Una vez que el número de elementos leídos es igual al intervalo de confirmación, el fragmento completo se escribe ItemWriter y luego se confirma la transacción. Este intervalo de confirmación se denomina chunk.

Parámetros de un step

- Reader: El ItemReader que proporciona elementos para su procesamiento.
- Writer: El ItemWriter que almacena los elementos proporcionados por el ItemReader.
- TransactionManager: comienza y confirma las transacciones durante el procesamiento.
- Repository: El JobRepository que almacena periódicamente StepExecution y ExecutionContext durante el procesamiento (justo antes de la confirmación).
- Chunk: Indica que este es un paso basado en el artículo y la cantidad de artículos que se procesarán antes de que se confirme la transacción.

B.2 Quartz

Quartz es un entorno de scheduling de código abierto que provee funcionalidad avanzada para la calendarización de tareas en Java.

Funcionalidades:

1. Cualquier tarea escrita en Java es susceptible de ser agendada dentro del entorno para ser ejecutada.
2. Permite agendar tareas en donde solo se indiquen la fecha y hora de ejecución y a partir de ahí definir una frecuencia de ejecución (ejemplo, ejecutar la tarea el primero de enero a las 12:00 a.m. y a partir de ahí, ejecutarse cada 3 días) hasta agendar tareas mediante el poder de las expresiones de Cron (ejemplo, ejecutar la tarea el segundo lunes de los meses de enero a septiembre del año 2008 a partir de las 8:00 a.m. cada 15 min. hasta las 10:00 a.m.).
3. Definir un medio de persistencia donde se almacene la información de tareas y sus agendas para poder darle la posibilidad al entorno de recuperar dicha información ante fallas de la aplicación.
4. Capacidad de trabajar en un ambiente de clusters.

Con esta funcionalidad, Quartz se trata de una excelente opción ya que proporciona un entorno robusto de scheduling, con integración en aplicaciones empresariales que requieran de alta disponibilidad mediante sus características de persistencia y clustering, adicionalmente ofreciendo la posibilidad de integrarlo con cualquier entorno sobre el cual se desarrolle la aplicación.

B.3 Spring Boot

Entorno para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Spring Boot permite un rápido inicio de una aplicación productiva. La idea de Spring Boot es que es muy fácil de ejecutar, por lo que minimiza el tiempo de configuración que conlleva la puesta en marcha de una aplicación. Se basa en dependencias y anotaciones. Esto simplifica la configuración, que es enorme para cuando un proyecto está creciendo y comienza a

tener un montón de dependencias para administrar. Todo está autoconfigurado. Utilizado para el lado del servidor de la aplicación de planificación de trabajos periódicos.

B.4 PostgreSQL

Es un potente sistema de base de datos relacional de objetos de código abierto con más de 30 años de desarrollo activo que le ha valido una sólida reputación por su fiabilidad, robustez y rendimiento. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). Esta tecnología se ha utilizado para el almacenamiento de la información de los trabajos periódicos.

B.5 MongoDB

Base de datos de documentos. Utilizada por la parte Backend para almacenar los datos como documentos JSON (Notación de objetos de JavaScript). Esta es una de las bases de datos no relacionales más famosas que hay. Integrar Mongo y JavaScript es super sencillo con la ayuda de mongoose, una biblioteca que nos permite modelar los esquemas de la base de datos de manera fantástica. Utilizada para el almacenamiento de los recursos de información extraídos.

B.6 Node JS

Desarrollado principalmente en JavaScript, permite implementar el Backend de la aplicación, utiliza un modelo asíncrono de E/S sin bloqueo controlado por eventos y de un solo thread. Esto lo hace increíblemente eficiente y liviano. Perfecto para aplicaciones muy intensivas en datos que necesitan operar en tiempo real en equipos distribuidos.

B.7 Express

Librería de Node.js que permite crear una infraestructura sólida para tu web, con él puedes manejar todo el Backend mediante rutas, orquestar todo el manejo de Webpack, logs y errores.

B.8 React

Es una biblioteca JavaScript de código abierto para construir interfaces de usuario. Es mantenida por Facebook y una comunidad de desarrolladores y empresas individuales. React se puede utilizar como base en el desarrollo de aplicaciones web o móvil de una manera limpia, organizada y permitirá que la interacción entre usuarios y la aplicación sea mucho más efectiva. Utilizada para la interfaz de usuario de la aplicación de presentación de resultados.

B.9 Angular JS

Es un entorno de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. Ejecuta su código JavaScript en el navegador del usuario, permitiendo que la UI de la aplicación sea dinámica. Utilizada para la interfaz de usuario de la aplicación de planificación de trabajos.

B.10 Docker y Docker-Compose

Se trata un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux para permitir que contenedores independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

Docker-Compose básicamente define cada uno de los contenedores que se quiere implementar, además de ciertas características para cada implementación del contenedor. Describe explícitamente cómo quiere implementar la aplicación de varios contenedores en el archivo docker-compose.yml. Una vez que se tiene este archivo de descripción de la implementación de varios contenedores, se puede desplegar el sistema completo en una sola acción organizada por el comando de la consola *docker-compose up*

El uso de docker-compose es básicamente un proceso de tres pasos:

1. Definir el entorno de tu aplicación con un Dockerfile para que pueda ser reproducido en cualquier lugar.
2. Definir los servicios que componen tu aplicación en docker-compose.yml para que se puedan ejecutar juntos en un entorno aislado.
3. Ejecuta docker-compose up y para inicia y ejecuta todo el sistema.

Anexo C Herramientas de gestión

C.1 Visual Studio Code

Herramienta de edición de código. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, ya que permite añadir extensiones para agregar nuevos idiomas, temas, depuradores y para conectarse a servicios adicionales. Las extensiones se ejecutan en procesos separados, lo que garantiza que no ralentizará su editor.

C.2 MongoDB Compass

Es una herramienta de interfaz gráfica para mostrar información sobre una base de datos de MongoDB y realizar consultas. Permite explorar visualmente tus datos, ejecutar consultas ad hoc en segundos, interactuar con sus datos con la funcionalidad completa de CRUD y optimizar el rendimiento de sus consultas.

C.3 Discord y Skype

Servicios utilizados para la comunicación, seguimiento y videoconferencias.

C.4 Google Drive

Se ha utilizado este servicio para almacenar y mantener actualizados todos los documentos del proyecto (pila de producto, memoria, diagramas…).

C.5 Tableros de proyectos en Trello

Se ha utilizado esta herramienta para la gestión de tareas. Se ha creado un tablero y se han establecido tres columnas: to do, in progress y done. En cada columna se ha ido posicionando las diferentes tareas según su progreso.

Anexo D Web Scraping

Web scraping es el proceso de recopilar información de forma automática de la web. Está muy relacionado con la indexación de la web, la cual indexa la información de la web utilizando un robot y es una técnica universal adoptada por la mayoría de los motores de búsqueda. Sin embargo, el Web scraping se enfoca más en la transformación de datos sin estructura en la web (como el formato HTML) en datos estructurados que pueden ser almacenados y analizados.

En el ámbito del proyecto, antes de cada trabajo de extracción, dada una url de la web donde se encuentran los ficheros que se desean obtener, se produce la descarga de todos estos ficheros. Para ello, se ha obtenido el código html de la página web y se ha realizado una búsqueda en ese código empleando expresiones regulares para encontrar el nombre de los ficheros que se desean descargar. Para ello, ha sido muy útil dominar regex el lenguaje de las expresiones regulares para delimitar las búsquedas o hacerlas más precisas y que el filtrado de la información sea mejor.

Una vez que se obtiene una lista con todos los ficheros encontrados en la web, se comprueba que cada fichero no se haya descargado antes. Si el fichero no se encuentra en el almacenamiento de la máquina se procede con su descarga y extracción.

En Java, se puede usar las clases *URL* y *HttpURLConnection* en el paquete *java.net* para descargar mediante programación un archivo desde una URL dada siguiendo estos pasos:

1. Crear un objeto URL para una URL determinada. La URL puede ser un enlace directo que contiene el nombre real del archivo al final, por ejemplo: https://www.mapa.gob.es/es/estadistica/temas/publicaciones/informesemanaldecoyunturas-23_tcm30-539769.xlsx
2. Abrir la conexión en el objeto URL, que devolverá un objeto HttpURLConnection si la URL es una URL HTTP.
3. Abrir el flujo de entrada de la conexión abierta.
4. Crear una secuencia de salida para guardar el archivo en el disco.
5. Leer repetidamente la matriz de bytes del flujo de entrada y escribirlos en el flujo de salida, hasta que el flujo de entrada esté vacío.
6. Cerrar el flujo de entrada, el flujo de salida y la conexión.

Tener en cuenta que se tiene que verificar el código de respuesta HTTP del servidor para asegurar de que la URL está disponible (estado HTTP 200) y luego se extrae el nombre del archivo de la propia URL. También se imprime información de depuración como Tipo de contenido, Disposición de contenido, Longitud de contenido y nombre de archivo. A la función implementada se le pasa como parámetros la url del archivo que se quiere descargar y el directorio destino.

Anexo E Almacén de certificados de Java

Es habitual encontrarnos en situaciones donde una empresa tiene su propia autoridad de certificación (Certificate authority o CA) con la que firma los certificados de sus servidores o bien nos encontramos con servicios REST o web services de sites cuyo certificado está firmado por una CA que no la tenemos en nuestro almacén de CAs de confianza.

En estos casos para que nuestras aplicaciones puedan establecer una conexión segura a estos servicios, necesitamos incorporar el certificado de la CA en nuestro almacén de CAs de confianza.

De esta forma, siempre y cuando nos encontremos con un certificado firmado por cualquiera de esas nuevas CAs, confiaremos automáticamente en ellos.

Como sabemos una CA firma los certificados que emite con su clave privada y es mediante el certificado de la CA con lo que podemos verificar la firma. Esta firma va incluida en el certificado emitido. Por ese motivo incorporamos el certificado de la CA entre los que confiamos.

En la Jdk el almacén de CAs de confianza lo vamos a encontrar en:

$JAVA_HOME/jre/lib/security/cacerts

Para incorporar el Certificado de la CA en el cacerts necesitamos importarlo. Se especifica la ruta del certificado y ruta del almacén de certificados java:

```
$JAVA_HOME/bin/keytool -import -noprompt -trustcacerts -alias InformeAgropecuario -file "E:\Universidad\4Cuarto\TFG\Proyectos\Harvesting-Infrastructure\src\main\resources\certificado.cer" -keystore "%JAVA_HOME%/lib/security/cacerts"
```

La contraseña por defecto del cacerts es: changeit.

Este comando nos pedirá confirmación de que queremos importar este nuevo certificado en el almacén de CAs.

Listas certificados:

```
$keytool -list -keystore "%JAVA_HOME%/jre/lib/security/cacerts"
```

Anexo F Diagrama de módulos

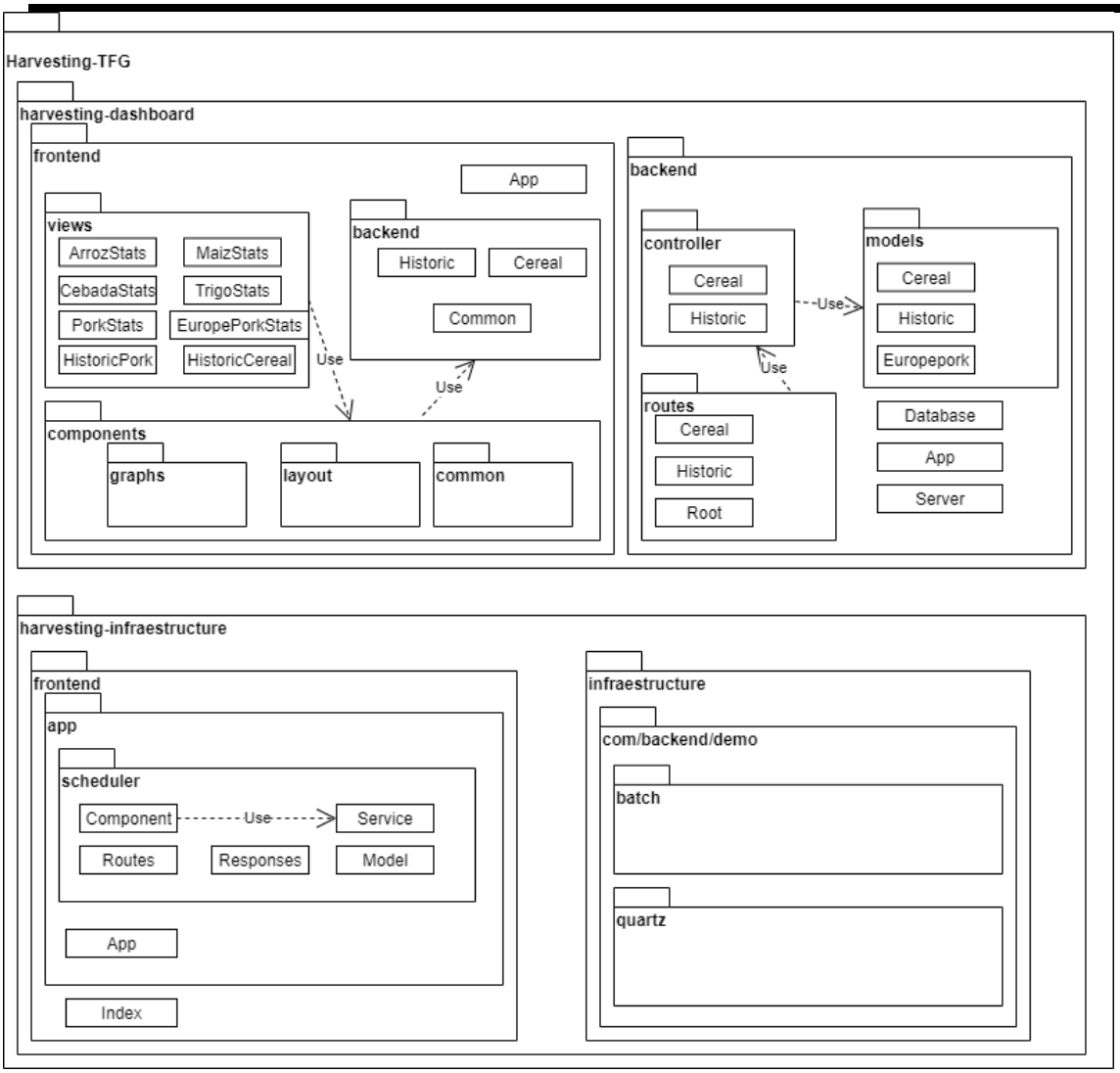

Figura 32: Diagrama de módulos del sistema

El sistema completo consta de dos módulos principales: harvesting-dashboard y harvesting-infraestructure.

Harvesting-dashboard: implementa la aplicación de presentación los resultados. Por un lado, el módulo frontend se encarga de construir las interfaces de usuario mediante vistas. Estas vistas se conforman de componentes que son los encargados de hacer las llamadas http a los diferentes servicios que ofrece el módulo Backend. Este módulo Backend desarrolla las diferentes rutas y controladores de la aplicación. Los controladores usan los modelos y acceden a la base de datos.

Harvesting-infraestructure: implementa la infraestructura de extracción de recursos y la aplicación de planificación de trabajos. El módulo frontend se encarga de construir las interfaces de usuario y realizar las llamadas http a la API que ofrece el módulo infraestructure.

El módulo infraestructure se divide en dos submódulos: Quartz y Batch.

Quartz implementa lo correspondiente a la gestión de los trabajos periódicos. Ofrece un controlador al cliente que hace uso de los diferentes servicios (crear, parar editar, eliminar un trabajo) y el módulo swagger documenta estos servicios.

Batch se encarga de implementar la extracción de datos mediante el módulo reader, el procesamiento con el módulo processor y carga de los diferentes recursos en una base de datos con el módulo writer. Además, este último hace uso del módulo repository para las llamadas a la base de datos.

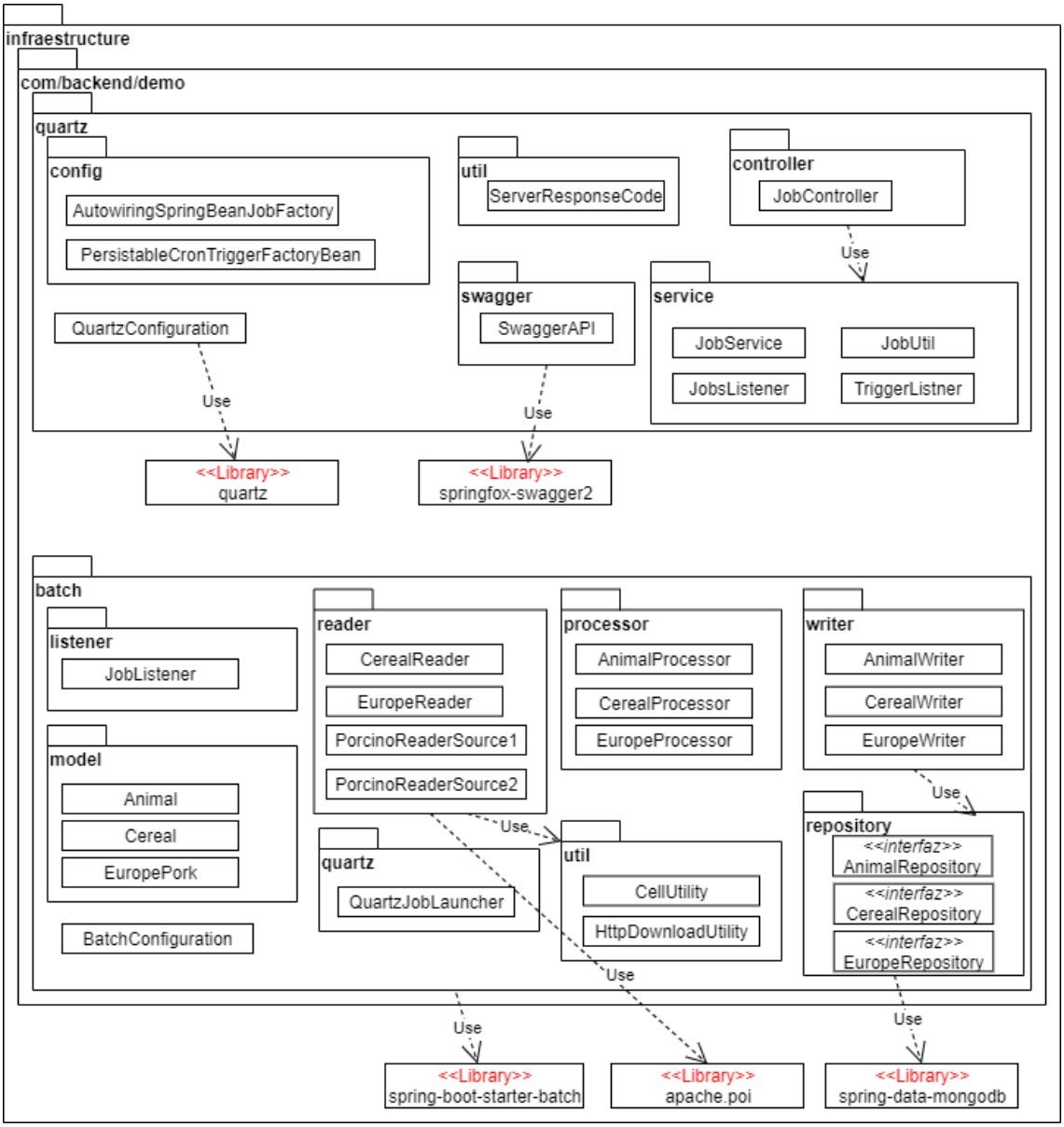

Figura 33: Diagrama del módulo infraestructura detallado

Anexo G Manual de Usuario

G.1 Aplicación web para la presentación de resultados

Porcino

Pantalla principal de la aplicación de presentación de resultados donde se muestran los datos extraídos del recurso porcino.

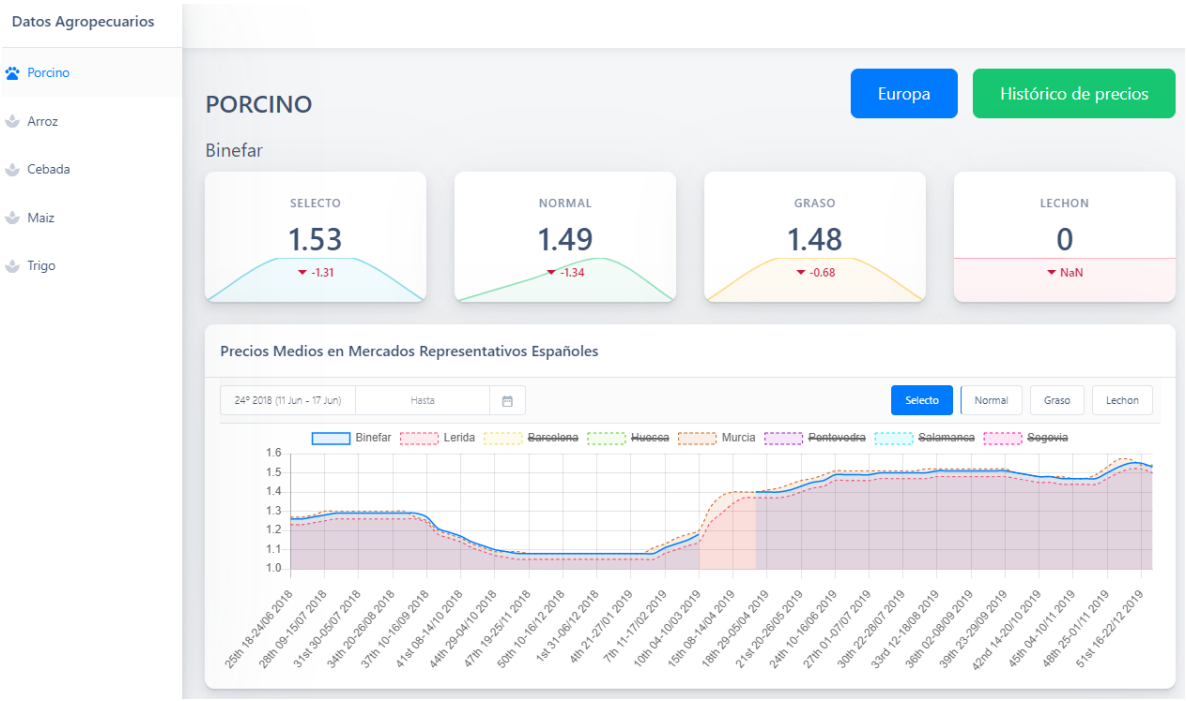

Figura 34: Pantalla principal de la web de presentación de resultados

Situada en la izquierda de la pantalla principal como se observa en la Figura 35, se encuentra una barra lateral desde la cual es posible seleccionar entre los distintos recursos agropecuarios que tiene la aplicación para visualizar su información.

Figura 35: Panel de categorías

A su vez, existe un componente descrito en la Figura 36, que está formado por dos botones que se encuentra en la parte superior derecha de la pantalla principal. El botón con la etiqueta Europa es para acceder a los precios del porcino en los mercados europeos y el botón con la etiqueta Histórico de precios es para visualizar el histórico de precios de ese recurso en España.

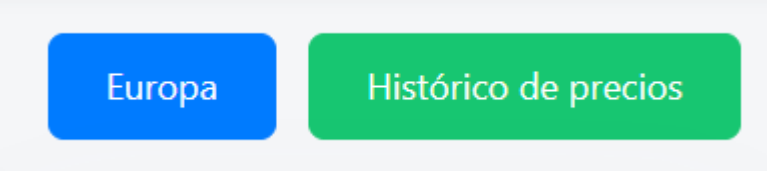

Figura 36: Botones de navegación

En la parte superior de la pantalla principal, como se ve en el componente de la Figura 37, es posible visualizar el precio de la última semana y la variación con respecto a la anterior y una gráfica temporal de los precios de las últimas cuatro semanas. Además, se produce una diferenciando según el tipo del recurso.

Figura 37: Cuadros de resumen

En la parte inferior de la pantalla principal, se encuentra una gráfica secuencial donde se muestra el precio en los diversos mercados a lo largo del tiempo, como se puede observar en la Figura 38. La gráfica permite acotar rangos entre dos fechas y agregar o eliminar la visualización de los mercados.

Figura 38: Gráfica temporal de precios y mercados

Pulsando en el botón Europa, el cual se observa en la Figura 36, aparece una nueva pantalla con una gráfica de los datos del porcino correspondientes a mercados europeos. En la gráfica se permite seleccionar el tipo, acotar rangos entre dos fechas y agregar o eliminar la visualización de los mercados.

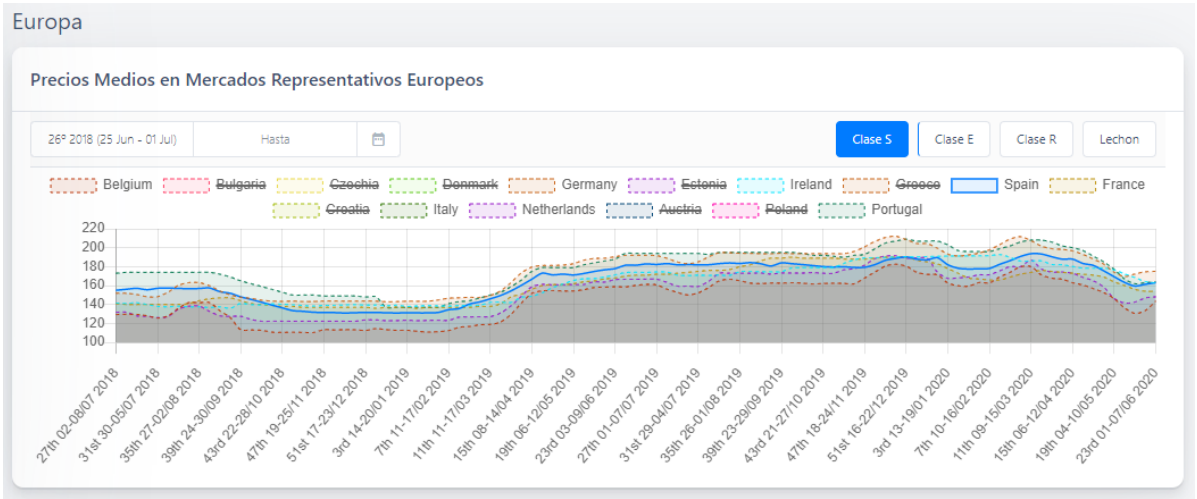

Figura 39: Gráfica del Porcino de mercados europeos

Pulsando en el botón de Histórico de precios, en los botones de navegación descritos en la Figura 36, aparece una nueva pantalla con las diferentes tablas para cada tipo donde se muestran los mercados y su precio en cada semana. Se permite la búsqueda y ordenar por fecha, precio de mayor a menor o viceversa.

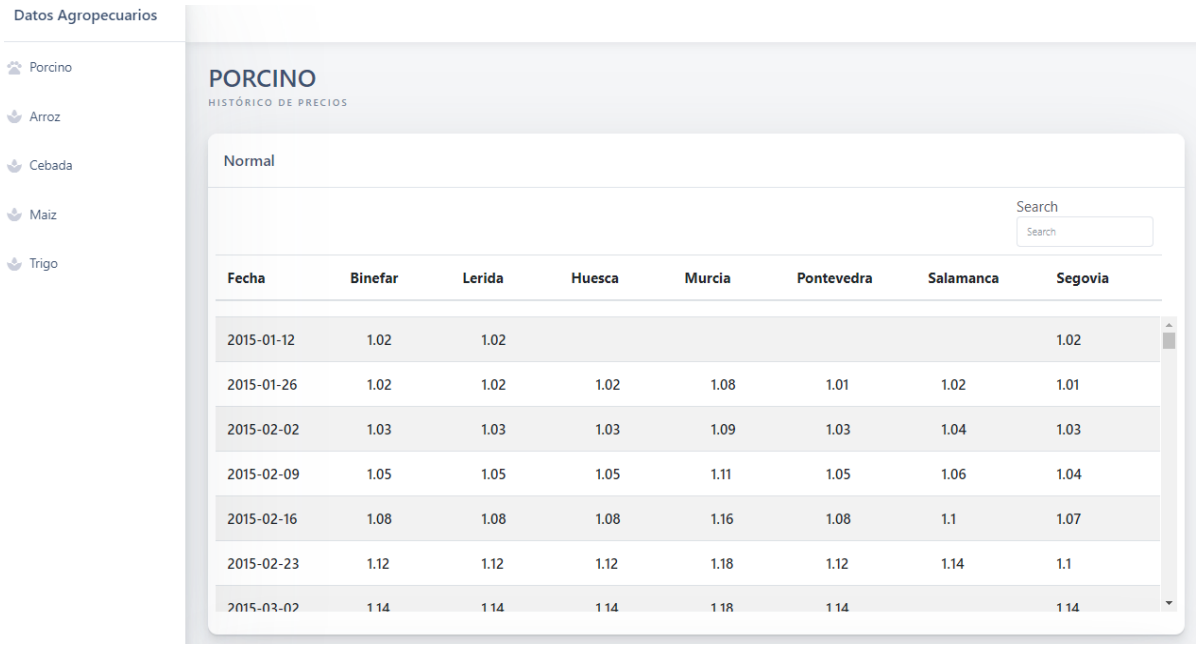

Figura 40: Histórico de precios del Porcino

Arroz

Seleccionando otro dato del panel de selección de categorías, se cambia a una pantalla distinta con la información de este nuevo recurso. La Figura 41 muestra la pantalla de información del arroz.

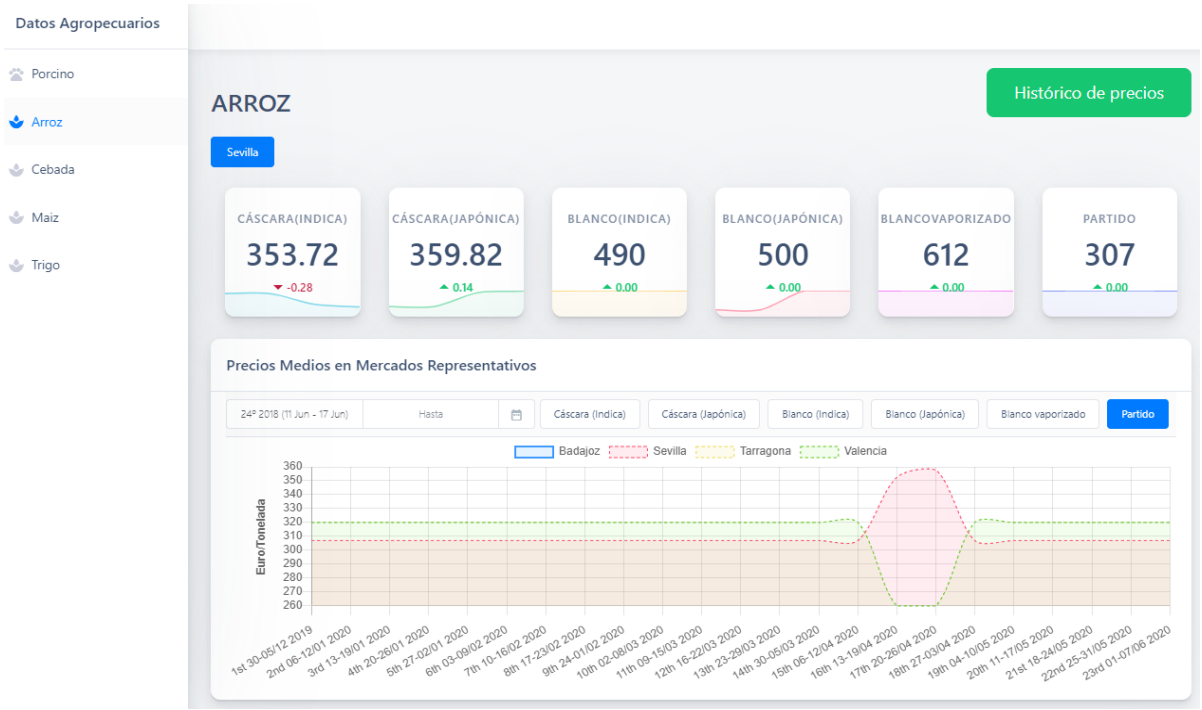

Figura 41: Datos del Arroz

Cebada

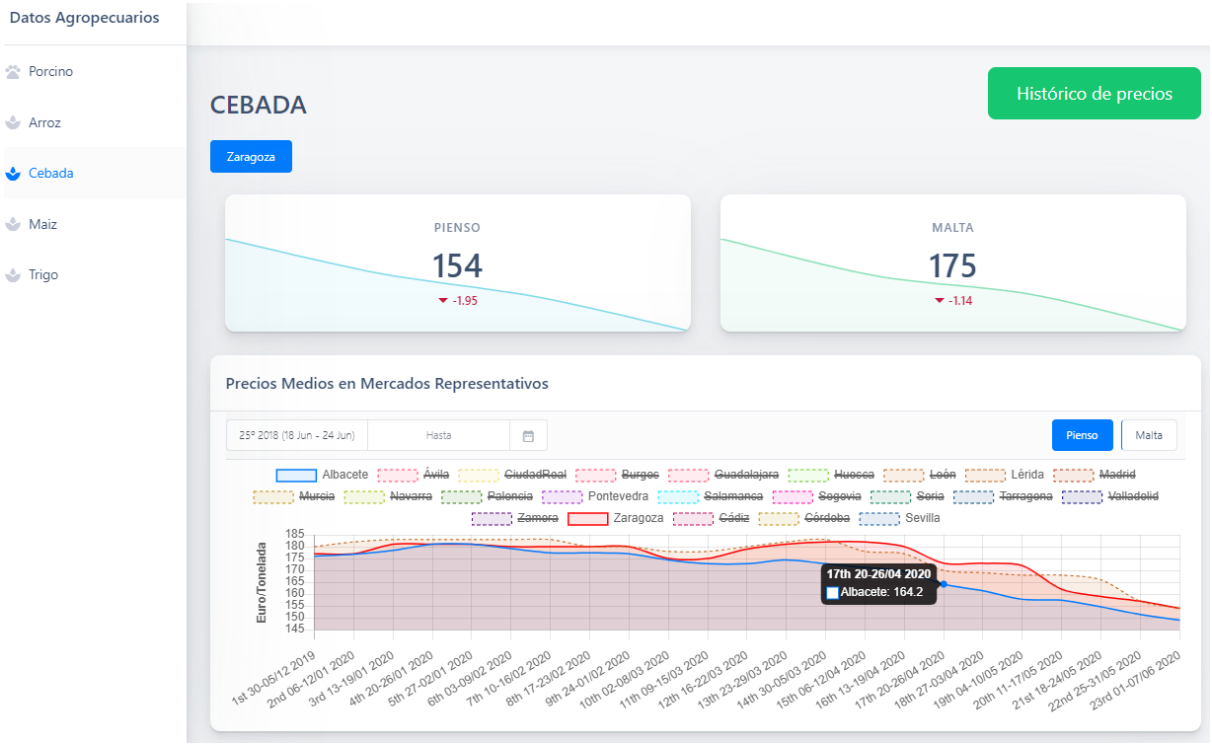

Figura 42: Datos de la Cebada

Maíz

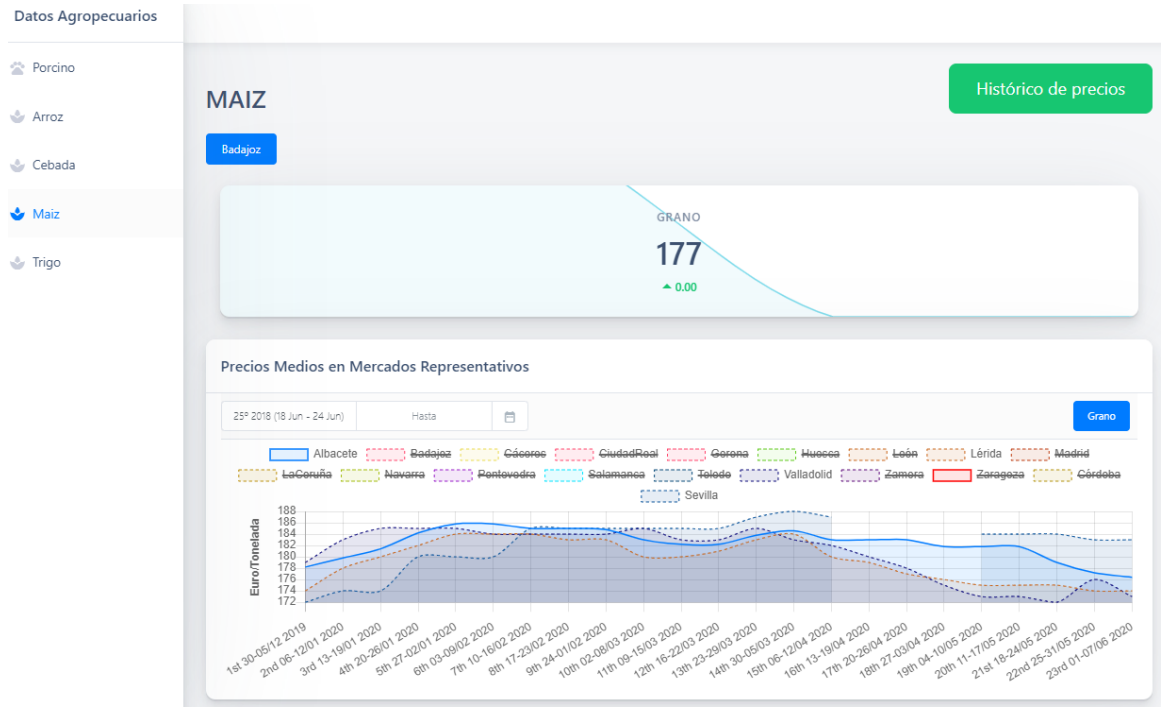

Figura 43: Datos del Maíz

Trigo

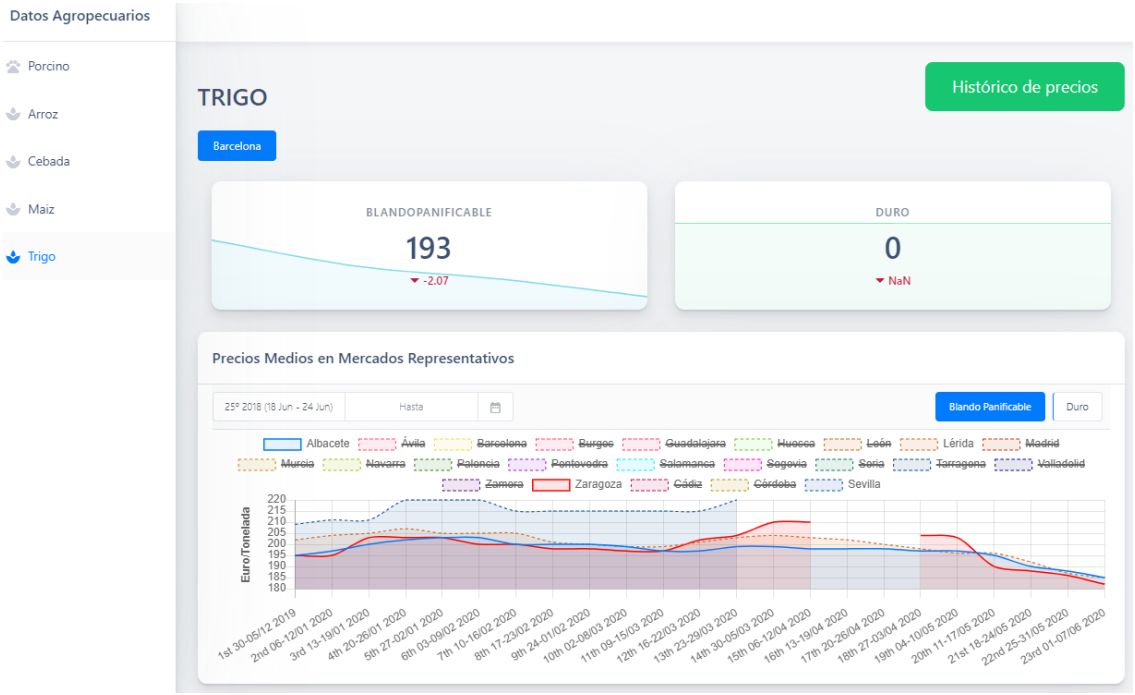

Figura 44: Datos del Trigo

G.2 Aplicación web para la planificación de trabajos ETL

Como se observa en la Figura 45, en la parte superior se permite crear un nuevo trabajo introduciendo un nombre y seleccionando el tipo de recurso que se quiere extraer (Porcino o Cereal). Para la ejecución temporal de este trabajo se permite introducir una fecha concreta o una expresión cron.

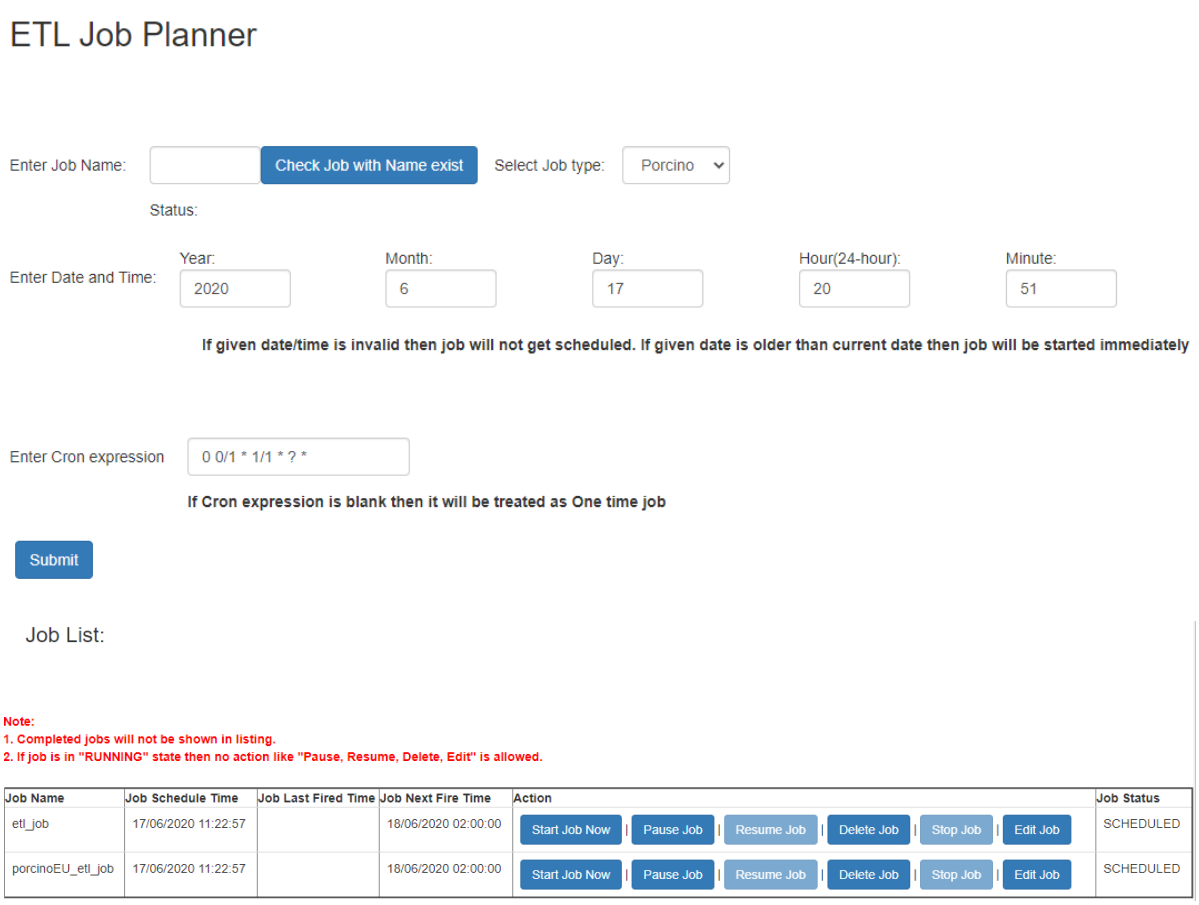

Figura 45: Aplicación web para la planificación de trabajos

En la parte inferior se visualiza una lista de todos los trabajos programados. Se muestra el nombre, fecha de creación, última fecha de ejecución, próxima fecha de ejecución, diferentes acciones y el estado actual del trabajo. Cada botón sirve para ejecutar una de estas acciones: Ejecutar en este instante, parar, reiniciar, eliminar, parar ejecución en marcha o editar información. Cada funcionalidad realiza una llamada HTTP a la API REST de servicios (API REST Planificación de trabajos, s.f.) que ofrece el Backend de esta aplicación.